The (Coarse) Fine-Grained Structure of NP-Hard SAT and CSP Problems

VICTOR LAGERKVIST*, Linköping University, Sweden
MAGNUS WAHLSTRÖM, Royal Holloway, University of London, Great Britain

We study the fine-grained complexity of NP-complete satisfiability (SAT) problems and constraint satisfaction problems (CSPs) in the context of the *strong exponential-time hypothesis* (SETH), showing non-trivial lower and upper bounds on the running time. Here, by a non-trivial lower bound for a problem $\mathrm{SAT}(\Gamma)$ (respectively $\mathrm{CSP}(\Gamma)$) with constraint language Γ, we mean a value $c_0 > 1$ such that the problem cannot be solved in time $O(c^n)$ for any $c < c_0$ unless SETH is false, while a non-trivial upper bound is simply an algorithm for the problem running in time $O(c^n)$ for some $c < 2$. Such lower bounds have proven extremely elusive, and except for cases where $c_0 = 2$ effectively no such previous bound was known. We achieve this by employing an algebraic framework, studying constraint languages Γ in terms of their algebraic properties. We uncover a powerful algebraic framework where a mild restriction on the allowed constraints offers a concise algebraic characterization. On the relational side we restrict ourselves to Boolean languages closed under variable negation and partial assignment, called *sign-symmetric* languages. On the algebraic side this results in a description via partial operations arising from system of identities, with a close connection to operations resulting in tractable CSPs, such as *near unanimity operations* and *edge operations*. Using this connection we construct improved algorithms for several interesting classes of sign-symmetric languages, and prove explicit lower bounds under SETH. Thus, we find the first example of an NP-complete SAT problem with a non-trivial algorithm which also admits a non-trivial lower bound under SETH. This suggests a dichotomy conjecture with a close connection to the CSP dichotomy theorem: an NP-complete SAT problem admits an improved algorithm if and only if it admits a non-trivial partial invariant of the above form.

CCS Concepts: • **Mathematics of computing** → **Discrete mathematics**; • **Theory of computation** → **Complexity theory and logic**;

ACM Reference Format:
Victor Lagerkvist and Magnus Wahlström. YYYY. The (Coarse) Fine-Grained Structure of NP-Hard SAT and CSP Problems. *ACM Trans. Comput. Theory* 1, 1, Article A (January YYYY), 53 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

The 3-SAT and k-SAT problems, of finding a satisfying assignment to a 3-CNF respectively k-CNF formula, are among the most well-studied NP-hard problems in terms of the precise computational complexity. We have seen a long range of improved algorithms for these problems, improving the trivial bound $O^*(2^n)$[1] to $O^*(c^n)$ for various constants $c < 2$. In particular for 3-SAT we have seen improvements from $O(1.6181^n)$ achieved in 1985 by Monien and Speckenmeyer [50] through to the latest improved bound of around $O(1.308^n)$ [26, 28, 29, 56], with a long list of papers in between.

*This is the corresponding author
[1]The notation O^* hides factors polynomial in the input size.

Authors' addresses: Victor Lagerkvist, Department of computer and information science, Linköping University, Linköping, Sweden, victor.lagerkvist@liu.se; Magnus Wahlström, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey, TW20 0EX, Great Britain, magnus.wahlstrom@rhul.ac.uk.

Particular milestones include the local search approach of Schöning [59] and the so-called PPSZ algorithm of Paturi, Pudlák, Saks and Zane [53]. However, the rate of improvement has been slowing down, and recent improvements for general 3-SAT have been reported to be on the order of the tenth digit after the decimal point [56]. Are we approaching a limit, or are there further breakthroughs ahead using new ideas? Similar questions can be asked about other problems, such as Maximum Independent Set [65], where we for a while saw improvement mainly in increasingly complex case analysis of branching schemes.

Clearly, we cannot take a slowing rate of improvement as direct evidence that further improvement is impossible. As an example, we may consider the problem Chromatic Number, where we saw several gradual improvements solving the problem in time $O(c^n)$, $c > 2.3$, until an ingenuous new idea gave an algorithm running in time $O^*(2^n)$ [8, Table 2]. Clearly, we would like some kind of an indication, even conjecturally, supporting a claim that, say, 3-SAT probably can't be solved in time $O(c^n)$ for some concrete value $c > 1$. But such claims have proven extraordinarily hard to find.

Two seminal hypotheses have been made in attempts to address these issues, the *exponential-time hypothesis* (ETH) and the *strong exponential-time hypothesis* (SETH) [14, 32]. For $k \geq 3$, let c_k denote the infimum of all constants $c > 1$ such that k-SAT can be solved in time $O(c^n)$. Then ETH is the statement that $c_3 > 1$, and SETH is the statement that $\lim_{k\to\infty} c_k = 2$, implying that no subexponential algorithms are possible for 3-SAT and that no exponential improvement over the trivial algorithm is possible for CNF-SAT in general.

Some results are known regarding the relative complexity of these problems. The seminal *sparsification lemma* [33] implies that ETH is equivalent to the statement that $c_k > 1$ for some constant k, and further research has shown that the sequence $(c_k)_{k=3}^{\infty}$ increases infinitely often assuming ETH [32]. Additionally, both ETH and SETH have been frequently treated as conjectures, and are used as indicators that some algorithms for problems in other domains may be optimal (e.g., "if you can solve this problem exponentially faster than our algorithm, then SETH is false"); see [19, 20, 47]. Still, despite all this, we are no closer to concrete lower bounds for c_k. Indeed, with some technical exception (see Example 3.15), the class of NP-hard SAT problems can be split into two parts: Problems which cannot be solved in time $O^*(c^n)$ for any $c < 2$ assuming SETH, implying that no exponential improvement over brute force is possible, and problems where no lower bound on the running time is known, except that $c > 1$ assuming ETH. In this article, we take steps towards changing this.

Our approach. To describe our approach, we need some definitions. We give summary descriptions here, with more technical background in the following subsection. A *constraint language* is a (possibly infinite) set of relations Γ. A *constraint* over Γ is defined by a relation $R \in \Gamma$ and a tuple $(x_1, \ldots, x_r)$ of variables, where r is the arity of R. It is *satisfied* by an assignment f if $(f(x_1), \ldots, f(x_r)) \in R$. We usually write $R(x_1, \ldots, x_r)$ or even $R(X)$ to represent such constraints. A relation is *Boolean* if it is over domain $\{0, 1\}$, and Γ is Boolean if every relation in Γ is Boolean. For a fixed Boolean constraint language Γ, the problem SAT(Γ) takes as input as set of variables V and a list of constraints over Γ with variables from V. The task is to find an assignment $f\colon V \to \{0, 1\}$ such that for every constraint $R(x_1, \ldots, x_r)$ is satisfied by f. The *constraint satisfaction problem* CSP(Γ) is the more general case when the domain is not necessarily Boolean.

For $k \geq 1$, let Γ_k denote the special language such that SAT(Γ_k) corresponds to k-SAT, i.e., Γ_k contains every Boolean relation R of arity $r \leq k$ such that $|R| = 2^r - 1$, since these relations correspond precisely to r-clauses. We then naturally extend the definition of c_k by defining $c(\Gamma)$ as the infimum over all constants $c > 1$ such that SAT(Γ) can be solved in time $O^*(c^n)$. Thus $c(\Gamma_k) = c_k$, and it is known that ETH implies $c(\Gamma) > 1$ for every language Γ such that SAT(Γ) is NP-complete. We give more background below.

The key to our approach is that we are studying the complexity of SAT(Γ) for languages Γ that are defined not directly, but only implicitly through their algebraic properties. Here, by an *explicit* definition we mean a language Γ that is defined either by exhaustive enumeration, or via a "structural description" of all relations $R \in \Gamma$. For example, SAT problems with explicitly defined languages Γ include every problem with a finite language, the usual SAT problem (CNF-SAT) defined via the language $\Gamma = \bigcup_{i=1}^{\infty} \Gamma_k$, and problems such as Horn SAT or even 0/1 Integer Programming, where the former is defined as allowing clauses with at most one unnegated variable, and the latter allows constraints of the type $\sum_i \alpha_i x_i = \beta$, with the restriction $x_i \in \{0, 1\}$.

Instead, we consider languages Γ containing all relations satisfying some algebraic property. The full background on this takes quite some preamble (see next subsection), but very briefly, it is known that the precise complexity of SAT(Γ), down to the value $c(\Gamma)$, is determined precisely by a set of algebraic invariants held by Γ, the *partial polymorphisms* of Γ [36]. These are algebraic properties that restrict the shape of the search space for instances of SAT(Γ). The larger the set of partial polymorphisms a language Γ observes, the more restricted is its search space, and consequently, the smaller the value $c(\Gamma)$ (although of course, this decrease in complexity is not necessarily strict). Conversely, if we want to study the most expressive languages Γ that still obey some restrictions to their expressive power, then we should consider languages Γ with as few partial polymorphisms as possible.

With this background, we proceed as follows. To make the question more manageable, we restrict our attention to *sign-symmetric* languages; informally, these are Boolean languages closed under variable negation and partial assignment. We then go through the following steps.

(1) We derive an algebraic characterization of the types of partial polymorphisms enjoyed by sign-symmetric languages. We refer to these as *pSDI-operations*. Among these, we characterize the *weakest* non-trivial pSDI-operations, corresponding to the richest, most expressive languages Γ, and undertake an initial investigation of their structure.

(2) We study the problems SAT(Γ) for languages $\Gamma = \mathrm{Inv}(f)$, defined as the language containing every relation R invariant under f, where f is one of the weakest non-trivial pSDI-operations just discussed. Therefore, these are the hardest possible problems SAT(Γ) which are not trivially SETH-hard, i.e., such that $c(\Gamma) < 2$ is possible under SETH. We find that, surprisingly, some of these problems correspond directly to classes of problems that can be solved by generic problem-solving strategies used for other SAT problems, such as the *meet-in-the-middle* and *local search* strategies. Thus, for some of these problems, we are able to show concrete bounds $c(\Gamma) < 2$.

(3) Finally, we study lower bounds on SAT(Γ) under SETH. Here, the richness of the language Γ comes in hand, and we are able to show a generic padding scheme that transforms an instance of SAT on n variables to an instance of SAT(Γ) for $\Gamma = \mathrm{Inv}(p)$ with $O(n)$ variables, where the constant depends only on the partial polymorphism f. Thus, for every language $\Gamma = \mathrm{Inv}(f)$ we study, we are able to show concrete bounds, saying that $c(\Gamma) \geq c_f$ under SETH, for some constant $1 < c_f < 2$.

Together, this provides the first problem SAT(Γ) such that we simultaneously have non-trivial upper bounds $c(\Gamma) < 2$ on its complexity, as well as a lower bound $c(\Gamma) \geq c_f > 1$ assuming SETH. However, the upper and lower bounds are still far apart, and there are many cases where the structure enforced by the pSDI-operation (if any) eludes us, where we are as of yet unable to determine the existence of an improved algorithm yielding $c(\Gamma) < 2$. We welcome further investigations into these matters; in particular, we leave open the question of a *dichotomy* characterizing every language Γ as either allowing $c(\Gamma) < 2$, or having $c(\Gamma) = 2$ under SETH.

1.1 The algebraic approach for fine-grained SAT and CSP complexity

In order to discuss our results and methods in more detail, let us provide some quick background on the algebraic notions used in the paper, and the *algebraic approach* to studying the complexity of CSPs.

Let Γ be a constraint language. The problems SAT(Γ) and CSP(Γ), as defined above, are referred to as the *parameterized satisfiability problem*, respectively the *parameterized constraint satisfaction problem*. (This has no connection to the field of parameterized complexity; it merely refers to the problem definition taking a "language parameter" Γ.) Studying the parameterized versions of these problems provides a broader perspective, compared to simpler approaches like studying languages defined only as allowing certain types of clauses, e.g. k-SAT and Horn-SAT.

Let us establish the terms *fine-grained complexity* of a language Γ to refer to the value $c(\Gamma)$ defined above, i.e., the precise time complexity of SAT(Γ) and CSP(Γ) up to subexponential factors. Conversely, the *coarse-grained complexity* simply refers to whether SAT(Γ) respectively CSP(Γ) is in P.

This coarse-grained complexity of SAT and CSP has been completely settled: for every finite language Γ, SAT(Γ) or CSP(Γ) is either in P or NP-complete, and we know the precise characterisation of the properties of Γ that decide this. For the Boolean case, this is Schaefer's classical dichotomy theorem [55], whereas the case of general finite domains (the *CSP dichotomy conjecture*) was more recently settled independently by Bulatov and Zhuk [11, 66] as the culmination of two decades of research.

Key to settling the CSP dichotomy conjecture was the algebraic approach taken for the problem. For an overview, see Bulatov [13] and Barto et al. [5] with further references; we briefly review the notions required for this paper.

The first observation is that the complexity of a problem SAT(Γ) or CSP(Γ) depends only on what we may informally call the *expressive power* of the language Γ. Between two languages Γ and Δ, if every relation of Γ can be "implemented" by relations from Δ, under some appropriate notion of implementation, then CSP(Δ) is at least as hard as CSP(Γ). The seed of the algebraic approach is then the fact that the expressive power of a language Γ can be characterized in purely algebraic terms through algebraic invariants of Γ using a *Galois connection*.

For coarse-grained complexity, a good starting point is the following.

Definition 1.1. Let Γ be a constraint language and R a relation. A *primitive positive definition (pp-definition)* of R over Γ is a formula F (i.e., a conjunction of constraints) over $\Gamma \cup \{=\}$, on a variable set $X \cup Y$, such that

$$R(X) \equiv \exists Y : F(X, Y).$$

We say that R is *pp-definable* over Γ if this holds. Similarly, a language Δ is pp-definable over Γ if every relation $R \in \Delta$ is pp-definable over Γ.

Such pp-definitions can be used as "gadgets" in problem reductions, and indeed, if Δ is pp-definable over Γ, then there is a polynomial-time reduction from CSP(Δ) to CSP(Γ) (and in fact also a logspace reduction) [13, 35]. The corresponding algebraic invariant is the following.

Definition 1.2. Let D be a finite domain and let $f\colon D^k \to D$ be an operation over D. Let $R \subseteq D^r$ be a relation over D. For tuples $t_1, \ldots, t_k \in D^n$, let $f(t_1, \ldots, t_k)$ be the result of applying f columnwise, i.e.,

$$f(t_1, \ldots, t_k) = (f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[n], \ldots, t_k[n])).$$

Then f is a *polymorphism of* R if and only if $f(t_1, \ldots, t_k) \in R$ for every sequence of tuples $t_1, \ldots, t_k \in R$. For a constraint language Γ, an operation f is a polymorphism of Γ if and only if it is a polymorphism of every relation $R \in \Gamma$. Synonymously to f being a polymorphism of R, we may say that f *preserves* R, or that R *is invariant under* f.

It is then known that for every pair of languages Γ, Δ, the language Δ is pp-definable over Γ if and only if every polymorphism of Δ is also a polymorphism of Γ [25, 35]. Hence, the coarse-grained complexity of SAT(Γ) and CSP(Γ) is indeed characterized by the polymorphisms of Γ.

However, for studying the fine-grained complexity the above characterization is insufficient, since the pp-definitions introduce additional existentially quantified variables Y. To characterize the fine-grained complexity of SAT(Γ) and CSP(Γ), let a *quantifier-free* pp-definition (*qfpp-definition*) of a relation R over a language Γ be a pp-definition that does not introduce any existentially quantified variables, i.e.,

$$R(X) \equiv F(X),$$

where $F(X)$ is a formula over $\Gamma \cup \{=\}$. The corresponding algebraic invariant is as follows.

Definition 1.3. Let D be a domain. A k-ary *partial operation* over D is a function $X \to D$ for some $X \subseteq D^k$. Let $f\colon X \to D$ be a partial operation over D and $R \subseteq D^r$ a relation over D. For $t_1, \ldots, t_k \in D^n$, let $f(t_1, \ldots, t_k)$ be the result of applying f columnwise, i.e., $f(t_1, \ldots, t_k)$ is undefined if there is at least one column $i \in [n]$ so that $f(t_1[i], \ldots, t_k[i])$ is undefined, otherwise $f(t_1, \ldots, t_k)$ is defined as in Def. 1.2. Then f is a *partial polymorphism* of R if and only if, for any sequence of tuples $t_1, \ldots, t_k \in R$, either $f(t_1, \ldots, t_k)$ is undefined or $f(t_1, \ldots, t_k) \in R$. Similarly, for a constraint language Γ, f is a partial polymorphism of Γ if and only if f is a partial polymorphism of R for every $R \in \Gamma$.

For a partial operation f, we use $\mathrm{Inv}(f)$ to denote the set of all relations preserved by f.

Generalizing the statement for polymorphism, a relation R has a qfpp-definition over a language Γ if and only if every partial polymorphism of R is also a partial polymorphism of Γ [54]. As a consequence, the fine-grained complexity of SAT(Γ) and CSP(Γ) is determined purely by the partial polymorphisms of Γ. In fact, for two languages Γ, Δ, if every partial polymorphism of Γ is also a partial polymorphism of Δ, then $c(\Delta) \leq c(\Gamma)$ [36]. In particular, if Γ is a Boolean language that has no non-trivial partial polymorphism, then $c(\Gamma_k) \leq c(\Gamma)$ for every k, thus $c(\Gamma) = 2$ under SETH.

Finally, we make a brief note on the algebraic approach in determining coarse-grained complexity of CSPs. For Boolean languages, studying pp-definitions generally suffices; e.g., Schaefer's dichotomy [55] can be conveniently stated in terms of the polymorphisms of Γ. However, for CSP(Γ) over larger domains, one generally uses a notion of implementation more powerful than pp-definitions; and correspondingly, a more general algebraic invariant. Again, see Bulatov [13] for a broader background. In particular, it is known that the coarse-grained complexity of CSP(Γ) depends only on the *identities* satisfied by polymorphisms of Γ (so-called *strong Maltsev conditions*) rather than upon the individual polymorphisms themselves. Such identities are equations satisfied by the polymorphism; and as we will see, there is a curious similarity between the partial polymorphisms we are led to study in this work and the identities characterizing some important classes of languages. For example, the following definitions are commonly used in the literature. Note that all these definitions refer to total operations, since they concern polymorphisms rather than partial polymorphisms.

- A *Maltsev operation* is any 3-ary operation $f\colon D^3 \to D$ over a domain D such that $f(x, x, y) = f(y, x, x) = y$ for any $x, y \in D$. Languages Γ preserved by a Maltsev operation generalize relations definable via linear equations, and the corresponding CSP can be solved in polynomial time for any such language [12]. For example, over a finite field F, the operation $f(x, y, z) = x - y + z$ is a Maltsev operation preserving solutions to systems of linear equations over F. In particular, Boolean relations preserved by the operation $f(x, y, z) = x \oplus y \oplus z$ are precisely the solutions to systems of linear equations over GF(2).

- A k-ary *near-unanimity (NU) operation* is any k-ary operation $f\colon D^k \to D$ over a domain D such that $f(y,x,\ldots,x) = f(x,y,x,\ldots,x) = \ldots = f(x,\ldots,x,y) = x$ for any $x,y \in D$, i.e., if all but one positions of f take the same value x, then f evaluates to x. It is known that if Γ has an NU operation of some arity $k \geq 3$, then CSP(Γ) can be solved efficiently using local consistency algorithms, e.g., propagation [5]. For the Boolean domain, the (uniquely defined) 3-ary NU operation is called *majority*, and a relation is preserved by majority precisely if it is the set of solutions to a 2-CNF formula.
- A *k-edge operation* is any $(k+1)$-ary operation f over a domain D such that for any $x,y \in D$, $f(y,y,x,x,x,\ldots,x) = f(y,x,y,x,x,\ldots,x) = x$, and $f(x,\ldots,x,y,x,\ldots,x) = x$ for any argument tuple over x,y containing precisely one y in position $i \geq 4$. This generalizes both previous cases, and precisely characterizes when a CSP can be solved (in polynomial time) by the *few subpowers* algorithm [7, 31].

1.2 Our results

We use the universal-algebraic toolkit presented in the previous section to study the fine-grained complexity of NP-hard problems SAT(Γ) and CSP(Γ). In particular, we study upper and lower bounds on $c(\Gamma)$ for Boolean languages $\Gamma = \mathrm{Inv}(p)$ defined by the existence of a single partial polymorphism f; by the discussion above, these are the richest languages Γ which could possibly admit an improved upper bound under SETH, i.e., for which $c(\Gamma) < 2$ is not immediately excluded by SETH.

To lead the discussion, call a Boolean language Γ *trivially SETH-hard* if Γ qfpp-defines all k-clauses for every $k \in \mathbb{N}$, i.e., for every $k \in \mathbb{N}$ and every $t \in 2^k$, Γ qfpp-defines the relation $R = 2^k \setminus \{t\}$. Similarly, call a partial operation f a *trivial polymorphism* if it is a polymorphism of every relation. It is easy to show the following:

- A partial r-ary operation f is a trivial polymorphism if and only if it is a subfunction of a projection, i.e., for some $i \in [r]$, $f(x_1,\ldots,x_r) = x_i$ for every tuple $(x_1,\ldots,x_r) \in 2^r$ on which f is defined.
- A language Γ is trivially SETH-hard if and only if every partial polymorphism of Γ is trivial.

Hence, the richest languages Γ which are not trivially SETH-hard correspond to languages $\Gamma = \mathrm{Inv}(f)$ for a single, non-trivial partial operation f.

To make the question more manageable, we restrict our attention to the natural class of *sign-symmetric languages*. As mentioned above, these are Boolean languages closed under variable negations and partial assignments; a more formal definition is given later in the paper (Def. 3.9). This represents a natural restriction that leads to a significant simplification of the algebraic framework, as we shall see. In addition, it is a natural restriction, which covers many SAT problems previously considered, including k-SAT, Exact SAT, and relations that can be modelled using bounded-degree polynomials [48] (see Theorem 3.1). Our investigations are presented in three parts: Algebraic and structural results, upper bounds on $c(\Gamma)$ (i.e., algorithms), and lower bounds under SETH. We now present these in turn.

Finally, we want to stress that the only mathematically arbitrary choice we make is the restriction to sign-symmetric languages (although we would of course argue that this choice is well justified). With this restriction in place, we simply consider the weakest possible partial polymorphisms characterizing sign-symmetric languages, in a mathematically precise sense, and investigate the languages $\Gamma = \mathrm{Inv}(f)$ and the problem SAT(Γ) for such languages. Therefore, we find it particularly interesting that several of these classes appear to coincide precisely with the language classes that can be solved more efficiently using common algorithmic strategies such as meet in the middle, fast matrix multiplication for CSPs, and local search for SAT.

Algebraic and structural results. We begin in Sections 3 and 4 by characterizing the algebraic invariants of sign-symmetric constraint languages. We show that every sign-symmetric language is characterized by partial polymorphisms f with the following properties:

(1) They are *self-dual*, i.e., for every tuple x such that $f(x)$ is defined, also its complement $f(\overline{x})$ is defined and $f(\overline{x}) = \overline{f(x)}$.
(2) They are *idempotent*, i.e., $f(x, \ldots, x) = x$ is defined for every $x \in \{0, 1\}$.

This result follows from previous work by the authors and Zanuttini [39, 43]. We call a partial operation which is self-dual and idempotent a *pSDI-operation*.

Due to the symmetries required, each pSDI-operation can be defined in a compact way using what we refer to as *polymorphism patterns*. These are partially defined equivalents of the identities of polymorphism used in coarse-grained CSP complexity, strong Maltsev conditions, as discussed above. Thus, in the total case, strong Maltsev conditions describe classes of operations which may result in tractability, and in the partial setting we obtain weaker, partial analogues of these operations, but which might still be sufficient to construct an improved exponential algorithm. For example, one of the cases studied in this article is the Boolean *partial Maltsev operation*, the Boolean operation f such that $f(y, x, x) = f(x, x, y) = x$ for any $x, y \in \{0, 1\}$, but $f(1, 0, 1)$ and $f(0, 1, 0)$ are undefined. We similarly define the Boolean *partial k-NU* operation for $k \geq 4$ as the k-ary operation f satisfying $f(y, x, \ldots, x) = f(x, y, x, \ldots, x) = \ldots = f(x, \ldots, x, y) = x$ for any $x, y \in \{0, 1\}$, with $f(x_1, \ldots, x_k)$ undefined otherwise. Each polymorphism pattern then naturally gives rise to a partial operation f_D over any domain D. Furthermore, if we "map" an n-ary relation R over $\{0, 1\}$ into a (n/d)-ary relation R' over the domain $D = 2^d$ in the natural way, and if R is preserved by a pSDI-operation f, then R' is preserved by the corresponding partial operation f_D. This connection is useful in some of the algorithmic results (see Section 5).

In order to study the richest languages $\Gamma = \mathrm{Inv}(f)$, we consider the weakest non-trivial pSDI-operations f, specifically those pSDI-operations f such that any pSDI-operation $f' \neq f$ attained by reducing the domain of f is trivial. We characterize all pSDI-operations that are minimal in this sense, and show that they are arranged in a hierarchy with clear *levels* according to their power. In particular we find the following:

- At the lowest level, there is a single pSDI-operation, the *partial Maltsev operation*.
- At every subsequent level, there is a unique strongest operation, the *partial k-NU operation*, as well as a unique weakest operation we refer to as the *k-universal operation*.

Other common algebraic identities, such as those defining *k-edge* operations also correspond to pSDI-operations in this hierarchy in a natural way.

We also investigate the structural consequences of these invariants, as well as the consequences of a language *not* being preserved by an invariant or a class of invariants; see Section 4 for details and examples. In particular, the language Γ_k of k-clauses is preserved by the partial $(k+1)$-NU operation, but not by any operation on an earlier level; and the language corresponding to roots of polynomials of degree at most k is preserved by the $(k+1)$-universal partial operation, but not by any stronger minimal operation. Furthermore, a sign-symmetric language Γ qfpp-implements all k-clauses *if and only if* it is not preserved by the k-universal partial operation.

Algorithms. Next, in Section 5 we investigate the possibility of non-trivial exponential-time algorithms for SAT(Γ) based only on the partial polymorphisms of Γ. For a pSDI-operation f, let Inv(f)-SAT denote the problem SAT(Γ) with

the only restriction being that every relation R used in an instance is preserved by f. For which minimal pSDI-operations f can this problems solved faster than 2^n, i.e., when is $c(\text{Inv}(f)) < 2$?

Although we do not manage to show this for every f, we show partial results, mainly the following.

- When f is the partial Maltsev operation, $\text{Inv}(f)$-SAT can be solved in time $O^*(2^{n/2})$ using a meet-in-the-middle strategy.
- When f is the 3-NU operation, then in the Boolean case, f is a total operation and $\text{Inv}(f)$-SAT is in P; however, over larger domains D, the corresponding problem of $\text{Inv}(f)$-CSP is NP-hard but solvable in time $O^*(|D|^{\omega n/3})$ using fast matrix multiplication, where $\omega < 2.373$ is the matrix multiplication exponent.
- When f is the partial k-NU operation, then we do not have a complete algorithm; but if the Erdős-Rado *sunflower conjecture* [22] holds for sunflowers with k sets, then $c(\text{Inv}(f)) < 2$ using a local search strategy.[2]

To the best of our knowledge, it is plausible that $c(\text{Inv}(f)) < 2$ for every non-trivial pSDI-operation f, but to actually show this involves some difficulty. For instance, consider the language whose relations can be modelled as the roots of bounded-degree multivariate polynomials over the real numbers, when evaluated at points $\{0,1\}^n$ only. This language is preserved by the $(d+1)$-universal partial operation, where d is the degree bound, but no algorithm is known for solving this with $c(\Gamma) < 2$ (see Definition 3.19 for a formal definition of the d-universal operation). The known algorithm for SAT with bounded-degree polynomials [48] (and its Boolean adaptation, cf. Theorem 3.1) relies upon the polynomial being over a fixed finite field.

The algorithms come with some moderately tricky representation issues, where some of the algorithms above work even if constraints are given only as oracles, while others need an explicit description. See Section 5 for details.

SETH-based lower bounds. In the final main contribution of the paper, we show in Section 6 that for every pSDI-operation f, there is a bound $1 < c_f < 2$ such that $c(\text{Inv}(f)) \geq c_f$ unless SETH fails. These are the first non-trivial lower bounds on $c(\Gamma)$ for any language Γ, except cases where $c(\Gamma) = 2$.

We show this result by appealing to the algebraic condition; we show that for any non-total pSDI-operation f there exists a universal "padding strategy", whereby any constraint $R(X)$ on $|X| = n$ variables can be padded to a constraint $R'(X,Y)$ such that $|Y| = O(n)$, $R(X) \equiv \exists Y R'(X,Y)$, and R' is preserved by f. Furthermore, the padding can be performed in a "reusable" way, so that distinct relations $R_1(X_1)$, $R_2(X_2)$ can reuse shared padding variables. See Sections 2.3 and 6 for details.

Our results give the following lower bounds on $c(\text{Inv}(f))$ assuming SETH:

- For the partial Maltsev operation, $c(\text{Inv}(f)) > 1.1547$.
- For the partial 3-edge and 3-universal operations, $c(\text{Inv}(f)) > 1.2599$.
- For the partial k-NU operation nu_k at $k \geq 4$, we show $c(\text{Inv}(\text{nu}_k)) \geq 2^{1-\frac{\log_2(k+1)}{k-1}}$. In particular, this is the strongest minimal pSDI-operation that is a partial polymorphism of $(k-1)$-SAT.

The best convergence to $c(\Gamma) = 2$ we are able to show for the partial k-NU operation, as a function of k, is at the rate $2 - O(\log k/k)$, whereas the convergence rate of the best known k-SAT algorithm is as $2 - O(1/k)$. As noted below, there is a conjecture, *Super-Strong ETH* (SSETH), which states that the slowest achievable convergence rate is $c_k = 2 - \Theta(1/k)$, but this has not been shown, even subject to other established conjectures (or even, strictly speaking, for the PPSZ algorithm in its most general form); see related work, below. If SSETH is to be connected to a more

[2]The best known bounds for the sunflower conjecture were recently drastically improved by Alweiss et al. [3]; unfortunately, barring anything but a complete resolution, our algorithm results remain conjectural.

established conjecture such as SETH, then it should be an easier first step to give evidence that $c(\mathrm{Inv}(\mathrm{nu}_k)) \geq 2^{1-O(1/k)}$, since $\mathrm{Inv}(\mathrm{nu}_{k+1})$-SAT is a vastly more expressive problem than k-SAT that still shares some of the structure of k-SAT.

1.3 Related work

SETH and related conjectures. ETH and SETH have become standard tools for conjectural lower bounds via reductions. In the current context of SAT, in addition to the foundational works of Impagliazzo et al. [32, 33] it is worth mentioning the lower bound for Not-all-equal SAT (NAE-SAT) and Hitting Set by Cygan et al. [19] and the lower bound for $\Pi_2$3-SAT by Calabro et al. [15]; an algorithm with a running time of $O(c^n)$ for one of these problems for some $c < 2$ would imply a faster algorithm for SAT, falsifying SETH. Many more such examples are known for parameters more permissive than n, e.g., for graph problems analysed for structural parameters such as treewidth [47].

Super-Strong ETH (SSETH) is the hypothesis that the running time of k-SAT depends on k as $c_k = 2 - \Theta(1/k)$, or more specifically, that running times of $c_k = 2 - f(k)/k$ are not possible for any unboundedly growing function $f(k)$ [60]. It seems fair to say that investigations of SSETH are still very early. It has long been known that the best known algorithms for k-SAT, including PPSZ, have upper bounds on their running time of the type $O(2^{(1-\Theta(1/k))n})$; Scheder and Talebanfard [57] recently provided a lower bound, showed that this behaviour is tight for (a mildly restricted version of) PPSZ. On the other hand, Vyas and Williams showed that random k-SAT formulas, which had been conjectured to be a very hard class, can be solved in time $O(2^{1-\Omega(\log k/k)n})$ [60]. This was improved by Lincoln and Yedidia to $O(2^{1-\Omega(\log^2 k/k)n})$ [46].

Fine-grained complexity of SAT and CSP. NP-hard SAT problems and CSPs with improved upper bounds (i.e., $O(c^n)$ for $c < 2$ for SAT, respectively $c < |D|$ for CSP) include k-SAT and simple sparse languages such as Exact SAT [61]. The most general case we are aware of is when relations are given implicitly as roots of bounded-degree multivariate polynomials over some finite field [48]. To be more precise, let $\mathbb{F}$ be a fixed finite field and $d \in \mathbb{Z}$ a degree bound, and consider the problem of finding a common root to a set of multivariate polynomials over $\mathbb{F}$ of degree at most d. Lokshtanov et al. [48] show that for any $\mathbb{F}$ and d, this problem can be solved in time $O(c_{\mathbb{F},d}^n)$ for some $c_{\mathbb{F},d} < |\mathbb{F}|$, where n is the number of variables. Note that in this case, the basic search space is simply $\mathbb{F}^n$; hence this is relevant to SAT only when $\mathbb{F} = GF(2)$. However, it is not hard to see that their algorithm can be adapted to the problem of finding a common root using only values 0 and 1 from $\mathbb{F}$ (i.e., a satisfying assignment $\phi\colon V \to \{0,1\}$) in time $O(c_{\mathbb{F},d}^n)$ for some $c_{\mathbb{F},d} < 2$; see Theorem 3.1 in Section 3.

In another direction, Brakensiek and Guruswami [9] show improved upper bounds for any optimization problem that can be given a particular form of LP relaxation, which implies improved algorithms for some CSPs. We note that despite similarities in definitions, their result does not apply to k-NU-SAT. Their result requires the existence of an infinite family of partial threshold functions as partial polymorphisms, where the operations of the family must have support of lower-bounded measure. While nu_k is a partial threshold function, nu_k alone does not imply the existence of such a family.

Non-Boolean examples also include the matrix multiplication-based algorithm for 2-CSP [62], the $O^*(2^n)$-time algorithm for k-Colouring [8], and Subset Sum [30]. The latter has recently been shown to have a polynomial-space algorithm with running time better than $O^*(2^n)$ by Bansal et al. [4].

Regarding lower bounds, it is known that $c(\Gamma) > 1$ assuming ETH for every NP-hard problem, both for Boolean [36] and non-Boolean languages [37]; i.e., neither problem type allows subexponential-time algorithms under ETH. Furthermore, curiously, there is an *easiest* NP-hard problem $\mathrm{SAT}(\Gamma_0)$ such that for every NP-hard problem $\mathrm{SAT}(\Gamma)$ we have

$c(\Gamma) \geq c(\Gamma_0)$ [36]. However, naturally, no concrete lower bound is known on $c(\Gamma_0)$ under SETH or any other plausible conjecture.

For other examples of "running time dichotomies" for SAT-like problems, see Bringmann et al. [10] and Künnemann and Marx [38].

Algebraic aspects of SAT and CSP complexity. Partial polymorphisms were first introduced to the CSP community by Schnoor & Schnoor [58] even though these notions were well-known in the algebraic community much longer [25, 54]. However, the principal motivation was to obtain dichotomy theorems for CSP-like problems incompatible with existential quantification, and the explicit connection to fine-grained time complexity of CSP was not realised until later by Jonsson et al. [36]. However, continued advancements in understanding this inclusion structure revealed that even severely restricted classes of constraint languages have a very complicated structure [18, 40]. In particular, it is known that any finite language Γ can be characterized only using an infinite number of partial polymorphisms [41]; in fact, for any finite set of purely partial polymorphisms F, the language $\Gamma = \mathrm{Inv}(F)$ contains $2^{2^{\Theta(n)}}$ relations of arity n, as opposed to finite languages Γ which can qfpp-define only $2^{n^{O(1)}}$ relations of each arity.

Another related topic is the investigation of *polynomial* (or linear) *kernels* for $\mathrm{SAT}(\Gamma)$ parameterized by n, also known as *sparsification*. The best known results are produced by phrasing constraints as roots of bounded-degree polynomials evaluated at $\{0,1\}^n$ [16], with an extension in terms of Maltsev languages which may or may not be more powerful [42]. Although there is no formal technical connection, variations on partial Maltsev polymorphisms occur as a possible success criterion for the latter. However, unlike the case of improved running times, it is known that no *finite* set of partial polymorphisms can suffice for polynomial sparsification [42]. Thus, in our earlier work [41, 42] we studied properties of invariant sets $\mathrm{Inv}(F)$, and properties of certain specific partial operations, but were unable to explicitly make the connection between these types of operations and improved running times.

Structure of the article. We begin the article in Section 2 with a self-contained example which illustrates the main ideas of our approach. Section 3 contains the necessary technical background, Section 4 characterizes sign-symmetric languages via pSDI-operations and describes their inclusion structure, and Section 5 and Section 6 tackle upper and lower bounds for sign-symmetric SAT and CSP problems. We conclude the article with a brief discussion in Section 7.

2 CASE STUDY: PARTIAL MALTSEV SATISFIABILITY

In this section, we pick one of the simplest invariants and provide self-contained upper and lower bounds on the running time of $\mathrm{SAT}(\Gamma)$ for the language Γ consisting of all relations preserved by this invariant. For the broader context and general definitions, see Section 3 and onward (as well as the relevant discussion in Section 1).

We focus on a problem we call *partial Maltsev satisfiability*, or Maltsev-SAT for short. This corresponds to the problem $\mathrm{SAT}(\Gamma)$ for the infinite language Γ consisting of all relations preserved by the partial operation ϕ, referred to as the *partial Maltsev operation*. The resulting language generalizes constraints definable via linear equations, such as Exact SAT and Subset Sum. (This language was also discussed in Section 1.2, although we provide a bit more context for the definitions below.)

For technical reasons, we assume that the constraints in the language are given via *extension oracles*. Concretely, let $R(X)$ be a constraint on a set of variables X. A *partial assignment* to X is an assignment $f\colon X_f \to \{0,1\}$ for some $X_f \subseteq X$. In the extension oracle model, we assume that $R(X)$ is provided in the form of an oracle that, given a partial assignment f to X, will reveal whether there exists an assignment $f'\colon X \to \{0,1\}$ such that f' satisfies $R(X)$ and

$f(x) = f'(x)$ for every $x \in X_f$. For now we assume that the oracle runs in polynomial time. A more general version will be given in Section 3.

Under this oracle model, we show the following:

(1) Maltsev-SAT on n variables can be solved in $O^*(2^{n/2})$ time, via a meet-in-the-middle strategy, and
(2) there is a constant $c(\phi) \sim 1.1547 > 1$ such that Maltsev-SAT cannot be solved in $O^*(c^n)$ for any $c < c(\phi)$ unless SETH is false.

But first, let us give a few more examples of relations preserved by ϕ.

2.1 Relations in $\mathrm{Inv}(\phi)$

Recall that a Maltsev operation on a domain D is any 3-ary total operation $f\colon D^3 \to D$ satisfying the identities

$$f(x, x, y) = f(y, x, x) = y$$

for any $x, y \in D$, with remaining evaluations $f(x, y, z)$, $x, y, z \in D$ taking some defined, but unspecified values in D. The *partial Maltsev operation* ϕ over the Boolean domain $D = \{0, 1\}$ is then, as discussed, the partial operation ϕ which is defined as above for inputs $x, y \in \{0, 1\}$ meeting one of the above two patterns, but with $f(1, 0, 1)$ and $f(0, 1, 0)$ undefined. So what languages are contained in the class $\Gamma = \mathrm{Inv}(\phi)$ discussed in this section? A complete satisfactory description of all such relations seems unlikely, for reasons we shall see soon, but we can give some illustrative examples.

As discussed, the corresponding total operation over $D = \{0, 1\}$ is the operation $f(x, y, z) = x \oplus y \oplus z$, and a relation $R \subseteq \{0, 1\}^n$ is preserved by f if and only if R is the set of solutions to a set of linear equations over GF(2). In other words, this language has a basis of parity relations $R \subseteq \{0, 1\}^n$ such that

$$R(X) \equiv \left(\sum X = b \pmod 2\right)$$

for $b \in \{0, 1\}$. Indeed, it is easily verified that if three tuples $t_1, t_2, t_3 \in \{0, 1\}^n$ satisfies such a constraint $R(X)$, then so does the tuple $t = t_1 \oplus t_2 \oplus t_3 = f(t_1, t_2, t_3)$. Since being preserved by the partial operation ϕ is less strict than being preserved by f, clearly $R \in \mathrm{Inv}(\phi)$ for every such relation. But that is a very small fraction of the language $\mathrm{Inv}(\phi)$.

For a broader class of examples, consider a (not necessarily finite) field $\mathbb{F}$ (or more generally a ring), and consider a linear equation

$$\sum_{i=1}^{n} \alpha_i x_i = \beta$$

where $\alpha_i, \beta \in \mathbb{F}$ are constants and the x_i are variables. Let $R_0 \subseteq \mathbb{F}^n$ be the set of solutions to this equation. Again, it is easy to verify that $f(x, y, z) = x - y + z$ is a polymorphism of R_0, since

$$\sum_{i=1}^{n} \alpha_i (x_i - y_i + z_i) = \beta - \beta + \beta = \beta$$

for any $x, y, z \in \mathbb{F}^n$ satisfying the equation. It is also easy to see that $f(x, y, z)$ is a Maltsev operation. Now, consider restricting this equation to only values $x \in \{0, 1\}^n$, i.e., let a relation R be defined as $R = R_0 \cap \{0, 1\}^n$. Then R is preserved by ϕ: for any $x, y, z \in R$ it holds that $x - y + z \in R_0$, and if $x_i - y_i + z_i \in \{0, 1\}$ for every $i \in [n]$, then also $x - y + z \in R$. In terms used in related work [42], R is a Boolean relation with a *Maltsev extension* R_0 over $\mathbb{F}$. Every such relation R also belongs to $\mathrm{Inv}(\phi)$.

This formulation provides some non-trivial NP-hard examples of instances of $\mathrm{Inv}(\phi)$-SAT. Exact SAT, as has been discussed, is the SAT problem where the question is whether an input CNF has a satisfying assignment where every

clause contains exactly one satisfied literal. In other words, the individual constraints of an Exact SAT instance can be written as

$$\sum_{j=1}^{r} \tilde{x}_{i_j} = 1$$

over $\mathbb{Z}$, where $\tilde{x}_{i_j} \in \{x_{i_j}, 1 - x_{i_j}\}$ for every j. Hence the constraints have Maltsev extensions over $\mathbb{Z}$ with coefficients $\alpha_i \in \{1, -1\}$ and with right-hand side $\beta \in \mathbb{Z}$.

For a more immediate example, if $\mathbb{F} = \mathbb{Z}$ and if a constraint $R(X)$ is provided only as the linear equation defining R_0, then it will in general be NP-hard even to test non-emptiness of a single constraint $R(X)$ (cf. Subset Sum). Hence, for such a language we can in general not construct an efficient extension oracle; however, we show in Theorem 6.3 that $\mathrm{Inv}(\phi)$-SAT is in general still at least as hard as Subset Sum in the extension oracle model. (See Section 3.2 for more on constraint representation.)

Finally, we claim that even this represents a small fraction of $\mathrm{Inv}(\phi)$, although it perhaps represents the extent of the *useful* portion of the language. Indeed, unlike most languages considered with explicit representations, the language $\mathrm{Inv}(\phi)$ is *dense*, in that there are $2^{2^{\Theta(n)}}$ relations $R \in \mathrm{Inv}(\phi)$ of arity n (cf. Section 4.1). Hence, beyond the manageable examples above, $\mathrm{Inv}(\phi)$ contains a large number of relations $R \subseteq \{0,1\}^n$ that appear to lack any useful structure, except that they are sufficiently sparse and irregular that any attempt to apply ϕ to R fails. Concretely, for any $x, y, z \in R$ there is an index $i \in [n]$ such that $\phi(x_i, y_i, z_i)$ is undefined. In this case, we say that ϕ applies *vacuously* to R. The lower bound presented in Section 2.3 exploits this phenomenon.

2.2 Upper bound

We now show the upper bound, i.e., that the language $\mathrm{Inv}(\phi)$-SAT can be solved in time $O^*(2^{n/2})$ via a meet-in-the-middle strategy. In fact, since exactly the same proof also works for relations over larger domains D, we show the more general result. That is, let D be a finite domain. Define the partial operation ϕ_D by letting $\phi_D(x, x, y) = \phi_D(y, x, x) = y$ for any $x, y \in D$, with $\phi_D(x, y, z)$ undefined in all other cases. Note that for $D = \{0, 1\}$ this definition coincides with the operation ϕ. We show that for any instance of $\mathrm{Inv}(\phi_D)$-CSP with constraints over the domain D, on n variables, we can test satisfiability in time $O^*(|D|^{n/2})$. The case $D = \{0, 1\}$ will then provide the promised improved algorithm for $\mathrm{Inv}(\phi)$-SAT.

As usual with exponential-time meet-in-the-middle strategies, we show an algorithm by splitting the variable set in two parts, enumerate the behaviour in each part exhaustively and locally, then using some global structure to connect the two halves to look for a satisfying assignment. We begin by setting the scene.

Let R be an n-ary relation on domain D, X a set of n variables, and $X = A \cup B$ a partition. We define a bipartite graph $H = H_R(A, B)$ as follows. The vertex sets of H are $U = D^A$ and $V = D^B$, i.e., the vertices in U are in bijection with the set of assignments $\alpha \colon A \to D$ and V is in bijection with assignments $\beta \colon B \to D$. Note that the edges uv of H where $u \in U$ and $v \in V$ are then in bijection with total assignments $X \to D$ to the variable set X. For assignments α to A and β to B, we let $\alpha \cup \beta$ denote the combined total assignment $X \to D$. Now let $H_R(A, B)$ contain an edge uv for $u \in D^A$ and $v \in D^B$ if and only if the corresponding combined assignment satisfies $R(X)$.

We note that for relations R preserved by ϕ_D, the graph $H_R(A, B)$ has a particularly nice structure, usually referred to as being *rectangular*. By tools presented in Section 5 this will follow immediately from general results, but in order to save on the amount of definitions needed for now, we show the property directly.

Lemma 2.1. *If R is a relation over D preserved by ϕ_D, then $H_R(A, B)$ is a vertex-disjoint union of bicliques.*

Proof. We first note that $H = H_R(A, B)$ is P_4-free, i.e., does not have the path on 4 vertices as an induced subgraph. Assume that there is a path $u_1 - v_1 - u_2 - v_2$ in H, and let the vertices u_i (respectively v_i) correspond to partial assignments α_i (respectively β_i), for $i = 1, 2$. Then the application

$$\phi(\alpha_1 \cup \beta_1, \alpha_2 \cup \beta_1, \alpha_2 \cup \beta_2) = \alpha_1 \cup \beta_2$$

is defined. Indeed, for every variable $a \in A$ the application of ϕ follows the pattern $\phi(a_1, a_2, a_2) = a_1$, and for every $b \in B$ the application follows the pattern $\phi(b_1, b_1, b_2) = b_2$. Hence the assignment $\alpha_1 \cup \beta_2$ also satisfies R and the edge u_1v_2 exists in H.

Hence H is a bipartite, P_4-free graph, and it is well-known, and not hard to see, that such graphs have the structure as described (for a modern reference see e.g. Chapter 9.3 in Fomin et al. [23]). Indeed, consider a connected component C of H, and let uv be a non-edge of C with $u \in U \cap V(C)$ and $v \in V \cap V(C)$. Let P be a shortest path from u to v in C. Since H is bipartite, P has length at least 3, hence P contains an induced P_4. Thus, the non-edge uv cannot exist, and every connected component of H is a biclique. □

By this the structure $H = H_R(A, B)$ can be described much more succinctly with a labelling scheme. Let $C_1, \ldots, C_\ell$ be the connected components of $H_R(A, B)$, including components on a single vertex, and for every vertex $w \in V(H)$ let the *label* of w be the index $i \in [\ell]$ of the component containing w. Then there is an assignment satisfying $R(X)$ if and only if there are vertices $u \in U$, $v \in V$ such that u and v take the same label. Let a *labelling scheme* for $R(X)$ be a procedure that given a vertex $w \in V(H)$ returns a label $\ell(w)$ such that for any two vertices $u \in U$, $v \in V$, we have $\ell(u) = \ell(v)$ if and only if u and v are in the same connected component in H. Such a labelling scheme can be used to identify the satisfying assignments of the constraint $R(X)$. For example, if $R(X) \equiv (\sum_{i=1}^{n} p_i x_i = q)$ is a linear constraint, with some coefficients p_i, q, then a natural label for a partition $[n] = A \cup B$ would be $\sum_{i \in A} p_i x_i$ for the set D^A and $q - \sum_{i \in B} p_i x_i$ for the set D^B.

More generally, let $F = R_1(X_1) \wedge \ldots \wedge R_m(X_m)$ be an instance of Inv(ϕ)-CSP, with each constraint $R_i(X_i)$ provided as an extension oracle. Let $X = X_1 \cup \ldots \cup X_m$ be the variable set of F. We extend this labelling principle to find assignments satisfying F. First, for simplicity extend every constraint to have the same scope X, i.e., for every $i \in [m]$ define

$$R'_i(X) \equiv R_i(X_i),$$

i.e., $R'_i = R_i \times D^{X \setminus X_i}$ (up to a reordering of the arguments). It is easily verified that R'_i is also preserved by ϕ. Indeed, assume that for three tuples $t_1, t_2, t_3 \in R'_i$ we have $\phi(t_1, t_2, t_3) = t$ defined and $t \notin R'_i$. Let t'_j, $j = 1, 2, 3$ be the projection of t_j onto positions X_i. Then $\phi(t'_1, t'_2, t'_3) = t'$ is also defined, and since $t \notin R'_i$ it must hold by construction of R'_i that $t' \notin R_i$, contradicting that ϕ preserves R_i. Hence every constraint $R'_i(X)$ admits a labelling scheme as above.

Now, to find two partial assignments $\alpha \colon A \to D$ and $\beta \colon B \to D$ such that the total assignment defined by $\alpha \cup \beta$ satisfies F it suffices to find α and β such that α and β get the same labels in the constraint $R'_i(X)$ for every $i \in [m]$.

Furthermore, assume that we have an *efficient* labelling scheme, i.e., one which computes a label in polynomial time using access to an extension oracle. Then an algorithm for Inv(ϕ_D)-CSP in time $O^*(|D|^{n/2})$ follows from standard methods. Indeed, we simply let $X = A \cup B$ be a partition with $|A|, |B| \leq n/2 + 1$, exhaustively enumerate the two sets $U = D^A$ and $V = D^B$, and for every $w \in U \cup V$ we let the label of w be the tuple $\ell(w) = (\ell_1(w), \ldots, \ell_m(w))$, where $\ell_i(w)$ for $i \in [m]$ is the label given to w by the labelling scheme for the constraint $R'_i(X)$. Using standard data structures, e.g., a red-black tree [17], we then simply check whether $\{\ell(u) \mid u \in D^A\} \cap \{\ell(v) \mid v \in D^B\} \neq \emptyset$, which can be done in time $O^*(|U| + |V|)$ where the star hides factors polynomial in $n + m + |D|$ only.

We close this section by noting that such an efficient labelling scheme is possible, even in the extension oracle model.

Lemma 2.2. *Let $R(X)$ be a constraint over a domain D, preserved by ϕ_D, and let $X = A \cup B$ be a partition. If $R(X)$ is provided as an extension oracle, then there exists an efficient labelling scheme.*

Proof. Fix an ordering $d_1 < \ldots < d_{|D|}$ for $D = \{d_1, \ldots, d_{|D|}\}$ arbitrarily, and extend it to tuples over D via the lex-min ordering; i.e., for two tuples $t, t' \in D^r$ let $t < t'$ if for some $i \in [r]$ we have $t[i] < t'[i]$ and $t[j] = t'[j]$ for $1 \leq j < i$. Observe that for any partial assignment $f : X' \to D$ for some $X' \subseteq X$, we can use the extension oracle to test whether there exists a total assignment extending f and satisfying $R(X)$, and among all such assignment $f' : X \to D$ we can find one that is lex-min in this ordering. Indeed, this is possible by simply iterating over the free arguments of f, and iterating through the possible domain values $d_1, \ldots, d_{|D|}$ for each one. This takes no more than $O(n|D|)$ queries to the extension oracle.

Now, we describe the labelling schemes for A and for B. Fix two distinct labels **L** and **R** to be used for rejected partial assignments. For an assignment $\alpha : A \to D$, if α is rejected by the extension oracle let $\ell(\alpha) = \mathbf{L}$, otherwise compute $\ell(\alpha)$ in two steps. First let $\beta : B \to D$ be such that $\alpha \cup \beta$ is the lex-min assignment extending α and satisfying $R(X)$. Next, let $\alpha' : A \to D$ be such that $\alpha' \cup \beta$ is the lex-min assignment extending β and satisfying $R(X)$. We let the label of α be α'.

For $\beta : B \to D$, let $\ell(\beta) = \mathbf{R}$ if the extension oracle rejects β, otherwise let $\alpha : A \to D$ be such that $\alpha \cup \beta$ is the lex-min assignment extending β satisfying $R(X)$, and let $\ell(\beta) = \alpha$.

Since the lex-min ordering is a total ordering on U and V in $H_R(A, B)$, and since every connected component of $H_R(A, B)$ is a biclique, it should be clear that this is a valid labelling scheme for $R(X)$. □

By the above arguments, and specialising to $D = \{0, 1\}$, we get the following.

Theorem 2.3. *maltsev-SAT is solvable in $O^*(2^{\frac{n}{2}})$ time.*

Our of all the arguments above, the main 'active ingredient' is the observation that $H_R(A, B)$ is rectangular if $R \in \text{Inv}(\phi)$. Indeed, all further arguments build on this property in standard ways. Hence the 'rectangularity property' of the invariant ϕ maps directly to an improved algorithm for $\text{Inv}(\phi)$-SAT. The next challenge is to determine whether similar useful structural properties are fixed by all other pSDI-invariants, in ways that allow for improved algorithms. As we shall see in Section 5, we are unable to determine this in the fully general case, but for some further noteworthy invariants it appears to hold.

2.3 Lower bound (overview)

Next, we observe lower bounds on $c(\text{Inv}(\phi))$ under SETH. For this section, we leave the general case of finite domains D for the last section and focus again on the case $D = \{0, 1\}$.

In passing, we note that the algorithm of Theorem 2.3 is very similar to the fastest known algorithms for Subset Sum. Hence, since solving Subset Sum on n integers in time $O^*(c^n)$ for any $c < \sqrt{2}$ is a long open question, it appears unlikely that a faster algorithm can be produced for $\text{Inv}(\phi)$-SAT using purely abstract properties that also hold for Subset Sum. In Section 6.2, we show that this connection can be made concrete: if $\text{Inv}(\phi)$-SAT can be solved in time $O^*(c^n)$ for some $c < \sqrt{2}$ in the extension oracle model then Subset Sum can be solved in time $O^*(c^{n+o(n)})$ (Theorem 6.3).

However, since this task is not known to be SETH-hard, we need a different lower bound strategy for $\text{Inv}(\phi)$-SAT. This strategy will replicate into a lower bound $c(\Gamma) \geq c_0 > 1$ for *every* language $\Gamma = \text{Inv}(f)$ defined by a non-total pSDI-operation f, with appropriate adjustments to the concrete constructions. We show this in Section 6.

The idea is the following. Let $R \subseteq 2^n$ be a Boolean relation. Recall that ϕ is said to apply *vacuously* to R if, for every $t_1, t_2, t_3 \in R$ such that $\phi(t_1, t_2, t_3)$ is defined, we have $\phi(t_1, t_2, t_3) \in \{t_1, t_2, t_3\}$. In particular, this implies that for almost every triple $(t_1, t_2, t_3) \in R^3$ there is a coordinate i such that $\phi(t_1[i], t_2[i], t_3[i])$ is undefined.

Concretely, note that for every coordinate $i \in [n]$, if $\phi(t_1[i], t_2[i], t_3[i])$ is defined, then

$$\phi(t_1[i], t_2[i], t_3[i]) \in \{t_1[i], t_3[i]\}.$$

If we have $t_2[i] = t_3[i]$ for every $i \in [n]$, then indeed $\phi(t_1, t_2, t_3) = t_1$, and conversely if $t_1[i] = t_2[i]$ for every $i \in [n]$ then $\phi(t_1, t_2, t_3) = t_3$. However, if there are two coordinates i, j such that $t_1[i] = t_2[i] \neq t_3[i]$ and $t_1[j] \neq t_2[j] = t_3[j]$, then $\phi(t_1[i], t_2[i], t_3[i]) \neq t_1[i] = t_2[i]$ and $\phi(t_1[j], t_2[j], t_3[j]) \neq t_3[j]$, hence $\phi(t_1, t_2, t_3)$ could not be one of the inputs t_1, t_2, t_3. Hence, for every such triple there must exist a further coordinate k with $t_1[k] \neq t_2[k] \neq t_3[k]$, so that $\phi(t_1[k], t_2[k], t_3[k])$ is undefined.

We will show a reduction from CNF-SAT on variables X to Inv(ϕ)-SAT on $O(|X|)$ variables, such that ϕ applies vacuously to every relation of the output instance. This will be achieved by adding a set of *padding variables* Y to the instance, such that $Y = f(X)$ for an appropriate function f, and for every triple of tuples (t_1, t_2, t_3) over X such that $\phi(t_1, t_2, t_3) = t \notin \{t_1, t_2, t_3\}$ would be defined, there is a padding variable y such that $\phi(y(t_1), y(t_2), y(t_3))$ is undefined. Here, $y(t)$ for $t \in 2^X$ denote the value of the padding variable y given the assignment t to X.

Pushing this idea through requires to overcome two obstacles. First, we need to perform the padding in a way such that the same set Y can be used to pad every distinct relation $R_i(X_i)$ of the input. Indeed, using a fresh set of padding variables Y_i for every k-clause C_i of the input leads to an unmanageable number of newly introduced variables. Second, the padding must be done in a way that still allows us to define an efficient extension oracle for the padded relations.

Our answer is to compute a "padding set" for the full relation $R(X) = 2^n$; i.e., a set of variables Y such that for every $(t_1, t_2, t_3) \in (2^n)^3$ such that $\phi(t_1, t_2, t_3) = t \notin \{t_1, t_2, t_3\}$ is defined there is a padding variable $y \in Y$ that blocks the application of ϕ as above. This padding set can then be reused for any smaller relation $R_i(X_i)$ as well.

For the second obstacle, we define our padding variables by a set of XOR equations over the input, i.e, every variable $y \in Y$ is defined as $y = \bigoplus_{i \in S_y} x_i$ for some set $S_y \subseteq [n]$. Furthermore, since the input is a set of disjunctive clauses, to implement an extension oracle for a clause C_i on a set of variables X_i we only need to test that the variables X_i are not forced to take the one assignment that is forbidden by C_i.

Padding set. As outlined above, we wish to define a set of "padding sets" $S_y \subseteq X$ that blocks every non-projecting application $\phi(t_1, t_2, t_3)$ of ϕ. For this, we first note the structure of such applications. Consider a triple (t_1, t_2, t_3) of tuples from 2^n such that $\phi(t_1, t_2, t_3) = t$ is defined, and let P (resp. Q) be the set of positions $i \in [n]$ such that $t_1[i] = t_2[i] \neq t_3[i]$ (resp. $t_1[i] \neq t_2[i] = t_3[i]$). Note that for any $i \in [n] \setminus (P \cup Q)$ we have $t_1[i] = t_2[i] = t_3[i] = t[i]$; moreover, $t \notin \{t_1, t_2, t_3\}$ if and only if both P and Q are non-empty.

Furthermore, let $S \subseteq [n]$ and define a padding variable $y(X) = \bigoplus_{i \in S} x_i$. We claim that $\phi(y(t_1), y(t_2), y(t_3))$ is undefined for the triple (t_1, t_2, t_3) above if and only if $|S \cap P|$ and $|S \cap Q|$ are both odd. This is perhaps non-obvious, but it is easily verified. Indeed, we have $y(t_1) \neq y(t_2)$ if and only if $|S \cap Q|$ is odd, since those are precisely the positions that switch values between t_1 and t_2, and we have $y(t_2) \neq y(t_3)$ if and only if $|S \cap P|$ is odd for the same reason. In particular, we can think of S as blocking (or not blocking) the pair (P, Q) itself, as it either blocks or does not block all $2^{|P|+|Q|}$ triples (t_1, t_2, t_3) generating the same pair (P, Q).

It is now easy to create a padding set as a randomized construction. For every triple (t_1, t_2, t_3) that needs to be blocked we define the pair (P, Q) of disjoint subsets of $[n]$ as above, making (less than) 3^n distinct pairs in total, and for

every pair a random set $S \subseteq [n]$ blocks all corresponding triples with probability $1/4$. Hence after selecting $m = cn$ random subsets, the expected number of unblocked pairs is (slightly less than)

$$3^n \left(\frac{3}{4}\right)^{cn} = 1$$

for $c = (\log 3)/(\log(4/3)) \approx 3.82$. Hence there exists a padding set with cn variables Y, and by Markov's inequality using $cn + d$ sets for $d = O(1)$ gives a correct construction with probability $\Omega(1)$. Derandomized constructions with $|Y| = O(|X|)$ are possible, but we omit the details.

Hence, assume that we have a collection of m sets $S_i \subset X$, numbering $m = cn + O(1)$ in total, and a set of m padding variables Y, where $y_i(X) = \bigoplus_{j \in S_i} x_j$ for every assignment X. Define the padding relation

$$R_0(X, Y) = \bigwedge_{i \in [m]} \left(y_i = \bigoplus_{j \in S_i} x_j \right)$$

on variables $X \cup Y$.

Extension oracle. Let $C_i(X_i)$ be the relation corresponding to a clause in the input on a set of variables $X_i \subseteq X$, and let $R_i(X, Y) = R_0(X, Y) \wedge C_i(X_i)$. We show that we can define an efficient extension oracle for $R_i(X)$. This has two parts. First, the central relation $R_0(X, Y)$, which is simply a set of linear equations over GF(2). Hence, it is easy to verify for any partial assignment f to $X \cup Y$ whether $R_0(X, Y)$ has a solution extending f, and if so, which variables, if any, are fixed among X in the solution set. Subject to this, the latter information allows us to verify $C_i(X_i)$. Indeed, since $C_i(X_i)$ is satisfied by all but one assignment to X_i, at least one solution to $R_0(X, Y)$ will satisfy $C_i(X_i)$ unless all variables of X_i are already fixed to values which fail to satisfy the clause. The latter can clearly be efficiently detected.

SETH lower bound. The above describes a (randomized) procedure that reduces an arbitrary instance of CNF-SAT, on n variables, to an instance of Inv(ϕ)-SAT with $|X| + |Y| = (1 + c)n + O(1)$ variables, with a polynomial-time extension oracle provided for every relation in the output. Hence, Inv(ϕ)-SAT cannot be solved faster than time

$$O^*(2^{\frac{1}{1+c}n}) = O^*(1.1547^n)$$

unless SETH is false.

For the general outlook, we note that the only details above that need to be adapted to the particular partial operation ϕ is the construction of the padding set (or even just the analysis of the probability of successful padding). We also note that there is quite some distance between the lower bound of $c(\text{Inv}(\phi)) \geq 2^{1/4.82}$ and the upper bound of $c(\text{Inv}(\phi)) \leq 2^{1/2}$. It would be very interesting to patch this gap.

As illustrated in this section, the partial Maltsev operation ϕ turned out be a rather well-behaved partial function whose SAT problem (1) extends natural SAT problems such as satisfiability of linear equations over finite fields, and (2) admits both an improved exponential-time algorithm and a concrete lower bound. This is certainly a promising starting point, but is it possible to describe partial functions of a similar nature, and, crucially, how they relate to each? Hence, how does Maltsev-SAT fit into the broader landscape of SAT problems? Can *all* SAT problems simpler than CNF-SAT be described using partial polymorphism invariants? These are some of questions that one should have in mind when we now properly begin the technical part of the article.

3 PRELIMINARIES

Throughout, for $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$.

A k-ary *relation* over a domain D is a subset of D^k. If $t = (x_1, \ldots, x_k)$ is a k-ary tuple we for every $1 \leq i \leq k$ let $t[i] = x_i$, and if $i_1, \ldots, i_{k'} \in [k]$ we write $\mathrm{Proj}_{i_1,\ldots,i_{k'}}(t) = (t[i_1], \ldots, t[i_{k'}])$ for the *projection* of t on the coordinates $i_1, \ldots, i_{k'}$. This notation easily extends to relations and we write $\mathrm{Proj}_{i_1,\ldots,i_{k'}}(R)$ for the relation $\{\mathrm{Proj}_{i_1,\ldots,i_{k'}}(t) \mid t \in R\}$.

A set of relations is called a *constraint language*, or simply a *language*, and will usually be denoted by Γ and Δ. We will typically define relations either by their defining logical formulas or by their defining equations. For example, the relation $R_{1/3} = \{(0,0,1), (0,1,0), (1,0,0)\}$ may be defined by the expression $R_{1/3} \equiv (x_1 + x_2 + x_3 = 1)$. However, we will not always make a sharp distinction between relations and their defining logical formulas and will sometimes treat e.g. a k-clause as a relation. We write $\mathrm{ar}(R)$ for the arity of a relation R, and use the notation Eq_D to denote the equality relation $\{(x, x) \mid x \in D\}$ over D.

A k-ary Boolean relation R is said to be *totally symmetric*, or just *symmetric*, if there exists a set $S \subseteq [k] \cup \{0\}$ such that $(x_1, \ldots, x_k) \in R$ if and only if $x_1 + \ldots + x_k \in S$. For example, $R_{1/3}$ is totally symmetric as witnessed by the set $S = \{1\}$. Symmetric relations will prove to be useful since it is sometimes considerably simpler to describe the symmetric relations invariant under a partial operation.

3.1 The parameterized SAT and CSP problems

Let Γ be a Boolean constraint language. The *parameterized satisfiability problem* over Γ (SAT(Γ)) is the computational decision problem defined as follows.

Instance: A set V of variables and a set C of constraint applications $R(v_1, \ldots, v_k)$ where $R \in \Gamma$, $\mathrm{ar}(R) = k$, and $v_1, \ldots, v_k \in V$.

Question: Is there a function $f : V \to \{0, 1\}$ such that $(f(v_1), \ldots, f(v_k)) \in R$ for each $R(v_1, \ldots, v_k)$ in C?

The *constraint satisfaction problem* over a constraint language Γ (CSP(Γ)) is defined analogously with the only distinction that Γ is not necessarily Boolean. We write (d, k)-CSP for the CSP problem over a domain with d elements where each constraint has arity at most k.

Bounded-degree polynomials. Lokshtanov et al. [48] showed that, for any finite field $\mathbb{F}$ and degree bound d, there is an algorithm that checks for the existence of a common root to a set of multivarite polynomials over $\mathbb{F}$ of degree at most d in time $O(c^n_{\mathbb{F},d})$, for some $c_{\mathbb{F},d} < |\mathbb{F}|$; i.e., an improved algorithm for the corresponding CSP where the domain is $\mathbb{F}$. We observe that their result also gives an improved algorithm for the corresponding SAT variant, where the search space is restricted to $\{0, 1\}^n \subseteq \mathbb{F}^n$. The proof can be found in the appendix.

Theorem 3.1. *Let $\mathbb{F}$ be a fixed finite field and $d \in \mathbb{N}$ a degree bound. There is a randomized algorithm that checks whether a given system of multivariate polynomials over $\mathbb{F}$ of degree at most d has a common root in $\{0, 1\}^n$ in time $O(c^n_{\mathbb{F},d})$ for some $c_{\mathbb{F},d} < 2$, where n is the number of variables in the system.*

Lokshtanov et al. [48] also present a deterministic version, but we focus on the randomized version.

3.2 The extension oracle model

We have defined the SAT and CSP problems but so far not mentioned how instances are represented. For finite constraint languages this is typically not of great importance, but since we in this article are mainly concerned with infinite constraint languages, upper and lower bounds may in fact differ depending on the representation in question. We

consider three distinct representations of SAT and CSP instances which we now define in detail. The most restrictive notion is the *non-uniform model*. In this, we consider an infinite "base language" Γ, e.g., $\Gamma = \mathrm{Inv}(f)$ for a partial operation f, and consider finite slices $\Gamma' \subset \Gamma$. We say that SAT(Γ) admits an improved algorithm *in the non-uniform model* if there is a constant $c(\Gamma) < 2$ such that, for every finite $\Gamma' \subset \Gamma$ there is an algorithm for SAT(Γ') with a running time of $O^*(c(\Gamma)^n)$. In this model, the precise representation of constraints is irrelevant. This is the strongest model to show lower bounds in.

Second, more permissively, we may let each relation R occurring in a constraint $R(x_1, \ldots, x_k)$ be represented as a list of tuples. We call this the *explicit representation*. This is one of the most frequently occurring representation methods in the algebraic approach to CSP. However, it is inconvenient in many applications, since a relation may contain exponentially many tuples with respect to the number of arguments. We therefore finally consider a more implicit representation where each constraint is represented by a procedure which can verify whether a partial assignment of its variables is consistent with the constraint.

Definition 3.2. Let R be an n-ary relation over a set D. A computable function which given indices $i_1, \ldots, i_{n'} \in [n]$ and $t \in D^{n'}$ answers yes if and only if $t \in \mathrm{Proj}_{i_1,\ldots,i_{n'}}(R)$ is called an *extension oracle representation* of R.

Hence, given a constraint $R(x_1, \ldots, x_n)$ and a partial truth assignment $f : X \to D$, $X \subseteq \{x_1, \ldots, x_n\}$, the extension oracle representation can be used to decide whether f can be completed into a satisfying assignment of $R(x_1, \ldots, x_n)$. We will usually implicitly assume that the oracle is polynomial-time computable, but for our purposes, subexponential time suffices; cf. Section 6.2.

Example 3.3. For each $r \geq 3$ and $0 < k < r$ consider the relation $R^r = \{(x_1, \ldots, x_r) \in \{0,1\}^r \mid x_1 + \ldots + x_r = k\}$. Even though $|R^r|$ is exponential with respect to r it is not difficult to see that constraints over R^r can be implicitly represented by computing the weight of the given assignment.

Example 3.4. CNF-SAT can be succinctly represented in the extension oracle model. Consider e.g. a positive clause $(x_1 \vee \ldots \vee x_n)$ and a partial truth assignment f on $\{x_1, \ldots, x_n\}$ with respect indices $i_1, \ldots, i_{n'}$. We then answer yes if f can be extended to a model of the clause, and no otherwise. Slightly more formally, we answer no if f is a total (i.e., $\{i_1, \ldots, i_{n'}\} = [n]$) assignment where $f(x_1) = \ldots = f(x_n) = 0$, and yes otherwise.

3.3 Partial polymorphisms and quantifier-free primitive positive definitions

Let D be a finite set of values. A k-ary *partial operation*, or a *partial function*, f over D is a mapping $X \to D$ where $X \subseteq D^k$, and a k-ary *total operation*, or just *operation*, over D is a function $D^k \to D$. The set X is said to be the *domain* of f and we let $\mathrm{domain}(f) = X$ denote this set and $\mathrm{ar}(f) = k$ denote the arity of f. If f and g are two n-ary partial operations over D such that $\mathrm{domain}(g) \subseteq \mathrm{domain}(f)$ and $g(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ for every $(x_1, \ldots, x_n) \in \mathrm{domain}(g)$ then g is said to be a *subfunction* of g. For $n \geq 1$ the i-ary *projection*, $1 \leq i \leq n$, is the operation $\pi_i^n(x_1, \ldots, x_i, \ldots, x_n) = x_i$ and a *partial projection* is any subfunction of a total projection.

If R is an n-ary relation over D and f a k-ary operation over D we say that f is a *polymorphism* of R if $f(t_1, \ldots, t_k) = (f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[n], \ldots, t_k[n])) \in R$ for each sequence of tuples $t_1, \ldots, t_n \in R$. If this holds then will sometimes also say that R is *invariant* under f or that f *preserves* R. Similarly, if f in this context is a partial operation we say that it is a *partial polymorphism* of R, that R is *invariant* under f, or that f *preserves* R, if for any sequence of tuples $t_1, \ldots, t_k$ we either have that $f(t_1, \ldots, t_k)$ is undefined or $f(t_1, \ldots, t_k) \in R$. We let $\mathrm{Pol}(R)$ and $\mathrm{pPol}(R)$ be the set of all polymorphisms and partial polymorphisms of the relation R, respectively, and if Γ is a constraint language we let

$\mathrm{Pol}(\Gamma)$ and $\mathrm{pPol}(\Gamma)$ denote the set of (partial) operations preserving each relation in Γ. Dually, if F is a set of (partial) operations we let $\mathrm{Inv}(F)$ be the set of all relations invariant under F. The two operators $\mathrm{Inv}(\cdot)$ and $\mathrm{pPol}(\cdot)$ are related by the following *Galois connection*.

Theorem 3.5 ([25, 54]). *Let Γ and Δ be two constraint languages. Then $\Gamma \subseteq \mathrm{Inv}(\mathrm{pPol}(\Delta))$ if and only if $\mathrm{pPol}(\Delta) \subseteq \mathrm{pPol}(\Gamma)$.*

Each set of partial operations F then naturally induces a SAT problem $\mathrm{SAT}(\mathrm{Inv}(F))$ where each relation involved in a constraint is preserved by every partial operation in F.

Definition 3.6. For a set of partial operations F we write $\mathrm{Inv}(F)$-SAT (respectively $\mathrm{Inv}(F)$-CSP), for the problem $\mathrm{SAT}(\mathrm{Inv}(F))$ (respectively $\mathrm{CSP}(\mathrm{Inv}(F))$).

The applicability of partial polymorphisms in the context of fine-grained time complexity might not be evident from these definitions. However, sets of the form $\mathrm{Inv}(F)$, called *weak systems* or *weak co-clones*, are closed under certain restricted first-order formulas which are highly useful in this context. Say that a k-ary relation R has a *quantifier-free definition* (qfpp-definition) over a constraint language Γ over a domain D if $R(x_1, \ldots, x_k) \equiv R_1(\mathbf{x}_1) \wedge \ldots \wedge R_m(\mathbf{x}_m)$ where each $R_i \in \Gamma \cup \{\mathrm{Eq}_D\}$ and each $\mathbf{x}_i$ is a tuple of variables of length $\mathrm{ar}(R_i)$.

For any set of partial operations F it is then known that $\mathrm{Inv}(F)$ forms a weak system, and is therefore closed under taking qfpp-definitions. Hence, Theorem 3.5 can equivalently be stated as: Δ qfpp-defines each relation in Γ if and only if $\mathrm{pPol}(\Delta) \subseteq \mathrm{pPol}(\Gamma)$. With this interpretation the following theorem is then a straightforward consequence.

Theorem 3.7. *[36] Let Γ and Δ be two finite constraint languages. If $\mathrm{pPol}(\Gamma) \subseteq \mathrm{pPol}(\Delta)$ then there exists a polynomial-time many-one reduction from CSP(Δ) to CSP(Γ) which maps an instance (V, C) of SAT(Δ) to an instance (V', C') of SAT(Γ) where $|V'| \leq |V|$ and $|C'| \leq c|C|$, where c depends only on Γ and Δ.*

In particular this implies that if $\mathrm{CSP}(\Gamma)$ is solvable in $O^*(c^n)$ time and $\mathrm{pPol}(\Gamma) \subseteq \mathrm{pPol}(\Delta)$ then $\mathrm{CSP}(\Delta)$ is solvable in $O^*(c^n)$ time, too. We will now briefly describe the closure properties of $\mathrm{pPol}(\Gamma)$, which are usually called *strong partial clones*. First, if $f, g_1, \ldots, g_m \in \mathrm{pPol}(\Gamma)$ where f is m-ary and each g_i is n-ary, then the *composition*

$$f \circ g_1, \ldots, g_m(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

is also included in $\mathrm{pPol}(\Gamma)$. This operation will be defined on a tuple $(x_1, \ldots, x_n) \in D^n$ if and only if each $g_i(x_1, \ldots, x_n)$ is defined and the resulting application over f is defined. Second, $\mathrm{pPol}(\Gamma)$ contains every partial projection, which is known to imply that $\mathrm{pPol}(\Gamma)$ is closed under taking subfunctions (i.e., if $f \in \mathrm{pPol}(\Gamma)$ then every subfunction of f is included in $\mathrm{pPol}(\Gamma)$). If F is a set of partial operations we write $[F]_s = \mathrm{pPol}(\mathrm{Inv}(F))$ for the smallest strong partial clone containing F.

3.4 Sign-symmetric constraint languages and pSDI-operations

In this section we describe a property of constraint languages which occurs naturally in the context of SAT problems, which also results in a simplified functional description on the polymorphism side.

Definition 3.8. An *n-ary sign pattern* is a tuple s where $s[i] \in \{+, -\}$ for each $1 \leq i \leq n$.

If t is an n-ary Boolean tuple and s an n-ary sign pattern then we let t^s be the tuple where $t^s[i] = t[i]$ if $s[i] = +$ and $t^s[i] = 1 - t[i]$ if $s[i] = -$. Similarly, if R is a Boolean relation and s an n-ary sign pattern we by R^s denote the relation

$\{t^s \mid t \in R\}$. Last, for $1 \leq i \leq n$ and $c \in \{0,1\}$ we let

$$R_{i=c} = \mathrm{Proj}_{1,\ldots,i-1,i+1,n}(\{t \mid t \in R, t[i] = c\})$$

be the relation resulting from freezing the ith argument of R to c and removing it from the relation.

Definition 3.9. A Boolean constraint language Γ is said to be *sign-symmetric* if (1) $R^s \in \Gamma$ for every n-ary $R \in \Gamma$ and every n-ary sign pattern s and (2) $R_{i=c} \in \Gamma$ for every $c \in \{0,1\}$ and every $1 \leq i \leq n$.

Clearly, for every Boolean language Γ there exists a unique, minimal sign-symmetric language, namely the smallest language $\Delta \supseteq \Gamma$ where (1) $R^s \in \Delta$ whenever $R \in \Delta$ and $s \in \{-,+\}^{\mathrm{ar}(R)}$, and (2) $R_{i=c} \in \Delta$ whenever $R \in \Delta$, $1 \leq i \leq \mathrm{ar}(R), c \in \{0,1\}$.

Example 3.10. Let $R(x,y,z) \equiv x_1 \vee x_2 \vee x_3$ be the set of models of the positive 3-clause. Then the relations $R^{(+,+,-)}, R^{(+,-,-)}$, and $R^{(-,-,-)}$ are simply the set of models of the clauses $(x_1 \vee x_2 \vee \neg x_3)$, $(x_1 \vee \neg x_2 \vee \neg x_3)$, and $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Similarly, a relation of the form $R_{2=0}$ is the set of models of the clause $(x_1 \vee 0 \vee x_3)$, i.e., the 3-clause where the second argument is forced to 0. If we let Γ be the smallest sign-symmetric language containing $\{R\}$ then SAT(Γ) is nothing else than the 3-SAT problem in slight disguise, where we allow the two constants 0 and 1 to occur in constraints.

Before presenting the algebraic characterisation of sign-symmetric constraint languages, let us consider an example which illustrates how sign-symmetry affects the expressive power of Γ.

Example 3.11. Let Γ be a sign-symmetric constraint language containing the symmetric 8-ary relation $R = \{(x_1,\ldots,x_8) \in \{0,1\}^8 \mid x_1 + \ldots + x_8 \in \{3,5,7\}\}$. Let $R_0 = \{(0)\}$ and $R_1 = \{(1)\}$ be the two constant Boolean relations. Then Γ can qfpp-define the positive 3-clause $S(x_1,x_2,x_3) \equiv (x_1 \vee x_2 \vee x_3)$ by first qfpp-defining the relation

$$S'(x_1,x_2,x_3,x_4,x_5) \equiv R(x_4,x_1,x_1,x_2,x_2,x_3,x_3,x_5) \wedge R_0(x_4) \wedge R_1(x_5),$$

and since we assumed that Γ is sign-symmetric, it follows that S is qfpp-definable as well, since $S = S''_{4=0}$ for $S'' = S'_{5=1}$.

We will now see that sign-symmetric constraint languages can be described by a natural class of partial polymorphisms. As a shorthand we will sometimes denote the k-ary constant tuple $(d,\ldots,d)$ by d^k.

Definition 3.12. Let f be a Boolean partial operation. We say (1) that f is *self-dual* if $\overline{x} \in \mathrm{domain}(f)$ for every $x \in \mathrm{domain}(f)$ and $f(x) = 1 - f(\overline{x})$, where $\overline{x}$ denotes the complement of the tuple x, and (2) that f is *idempotent* if $d^k \in \mathrm{domain}(f)$ and $f(d^k) = d$ for every $d \in D$.

In the sequel, we will call a Boolean partial operation which is both self-dual and idempotent a *pSDI-operation*, short for partial, self-dual, and idempotent operation. Sign-symmetric constraint languages have a strong connection to pSDI-operations, formalised in the following theorem.

Theorem 3.13. *[39, 43] Let F be a set of pSDI-operation. Then:*

(1) $\mathrm{Inv}(F)$ *is sign-symmetric, and*
(2) *if* Γ *is a sign-symmetric constraint language such that* $\mathrm{Pol}(\Gamma)$ *contains only projections (meaning that SAT*(Γ) *is NP-complete) then every* $f \in \mathrm{pPol}(\Gamma)$ *is a subfunction of a pSDI-operation.*

Example 3.14. Recall the ternary partial operation ϕ from Section 2 defined via $\phi(a,a,b) = b$ and $\phi(a,b,b) = a$ for all $a,b \in \{0,1\}$, and undefined otherwise. Hence, to be precise:

(1) $\phi(0,0,0) = \phi(1,1,0) = \phi(0,1,1) = 0$, and
(2) $\phi(1,1,1) = \phi(0,0,1) = \phi(1,0,0) = 1$.

We conclude that ϕ is idempotent since $\phi(0,0,0) = 0$ and $\phi(1,1,1)$, and that ϕ is self-dual since in addition

(1) $\phi(1,1,0) = 1 - \phi(\bar{1},\bar{1},\bar{0}) = 1 - \phi(0,0,1) = 0$, and
(2) $\phi(0,1,1) = 1 - \phi(\bar{0},\bar{1},\bar{1}) = 1 - \phi(1,0,0) = 0$.

Thus, ϕ is a pSDI-operation, meaning that $\mathrm{Inv}(\phi)$ is sign-symmetric.

We remark that Theorem 3.13 was originally proved under a slightly different notion of sign-symmetry where constant arguments are not necessarily projected away. To be precise, Lagerkvist, Wahlström & Zanuttini [43] defined $R_{i=c}$ as $\{(x_1,\ldots,x_{i-1},c,x_{i+1},\ldots,x_n) \mid (x_1,\ldots,x_n) \in R\}$, meaning that every sign-symmetric constraint language under this definition is included in a sign-symmetric constraint language under our current definition. However, the proof remains valid under our current notion of sign-symmetry. Hence, the study of sign-symmetric constraint languages reduces to studying properties of pSDI-operations. Let us give two examples illustrating why focusing on sign-symmetric languages is indeed a simplification. First we demonstrate a language which admits improved algorithms but in a less interesting way.

Example 3.15. Consider a Boolean language Γ consisting of relations of the type

$$R(x_1,\ldots,x_{2r}) \equiv R'(x_1,\ldots,x_r) \wedge (x_1 \neq x_{r+1}) \wedge \ldots \wedge (x_r \neq x_{2r}),$$

for every Boolean relation R'. Then $\mathrm{SAT}(\Gamma)$ can be solved in $O^*(2^{\frac{n}{2}})$ time, simply due to the fact that there for every variable v in an instance exists at least one variable v' such that $v \neq v'$ in every solution. Hence the variable set of the instance is divided into clusters of at least two variables each. It is also clear that no faster algorithm is possible under SETH, since R' can be an arbitrary k-clause. Thus this language has (under SETH) a tight running time of $O^*(2^{\frac{n}{2}})$, but in an algorithmically uninteresting way.

This language has an infinite number of partial polymorphisms: any partial operation f that is undefined on t or $\bar{t}$ for every pair $(t,\bar{t})$ of complementary tuples of $\{0,1\}^{\mathrm{ar}(f)}$ is a partial polymorphism of Γ, simply because every attempt to apply f to tuples of some $R \in \Gamma$ will have at least one coordinate with an undefined value.

However, the sign-symmetric closure of Γ also contains the relation R'' where the second half of the positions of R are negated, meaning that

$$R''(x_1,\ldots,x_{2r}) \equiv R'(x_1,\ldots,x_r) \wedge \mathrm{Eq}_{\{0,1\}}(x_1,x_{r+1}) \wedge \ldots \wedge \mathrm{Eq}_{\{0,1\}}(x_r,x_{2r}),$$

which clearly yields an SETH-hard language with running time $O^*(2^n)$ only assuming SETH. To see this, simply note that the relation R'' can define R' by repeating variables, i.e., $R'(x_1,\ldots,x_r) \equiv R''(x_1,\ldots,x_r,x_1,\ldots,x_r)$.

Next, we demonstrate a language Γ with strongly restricted expressive power in a technical sense, but which still yields an SETH-hard language.

Example 3.16. Let Γ contain relations $R_{\geq 1}(x_1,\ldots,x_r) \equiv (x_1 \vee \ldots \vee x_r)$ and $R_{=k}(x_1,\ldots,x_r) \equiv (x_1 + \ldots + x_r = k)$ of all arities r and for all values k. Then Γ has strongly restricted expressive power, e.g., it cannot qfpp-define the negative 2-clause $(\neg x_1 \vee \neg x_2)$, even though it can pp-define this 2-clause via $(\neg x \vee \neg y) \equiv \exists z\colon R_{=1}(x,y,z)$. The fact that Γ cannot qfpp-define e.g. $(\neg x_1 \vee \neg x_2)$ can be evidenced by a partial polymorphism: define the partial operation f as $f(0,0,0) = 0$ and $f(1,0,0) = f(0,0,1) = f(1,1,1) = 1$. It is easy to verify that f preserves Γ but not any negative 2-clause. However, consider the following Turing reduction from the Hitting Set problem: given a hypergraph and a parameter $t \in \mathbb{N}$, for

each $1 \leq k \leq t$ create a SAT(Γ) instance where the hypergraph is translated to $R_{\geq 1}$ constraints in the natural way, and where we introduce a single $R_{=k}$ constraint which makes sure that only k variables are assigned the value 1. Hence, $c(\Gamma) = 2$ under SETH [19]. On the other hand, the sign-symmetric closure of Γ contains all k-clauses for all k. Hence, this problem is trivially SETH-hard.

3.5 Polymorphism patterns

In this section we describe a powerful method for constructing pSDI-operations with a natural correspondence to polymorphisms occurring in CSP classification projects. In fact, we will see that all pSDI-operations, and hence all sign-symmetric constraint languages, can be described in this form. We use this method to define several highly relevant classes of pSDI-operations which we will return to many times in the sequel.

Definition 3.17. Let a *polymorphism pattern* of arity r be a set of pairs (t, x) where t is an r-ary tuple of variables and where x occurs in t. We say that an r-ary partial operation f over a set of values D is defined by an r-ary polymorphism pattern P if

$$\text{domain}(f) = \{(\tau(x_1), \ldots, \tau(x_r)) \mid ((x_1, \ldots, x_r), x) \in P, \tau : \{x_1, \ldots, x_r\} \to D\}$$

and $f(\tau(x_1), \ldots, \tau(x_r)) = \tau(x)$ for every $((x_1, \ldots, x_r), x) \in P$ and every $\tau : \{x_1, \ldots, x_r\} \to D$.

Example 3.18. It is easy to see that a polymorphism pattern also induces a set of total operations where each operation satisfies the tuples in the polymorphism pattern, but may result in distinct values otherwise. For example, a ternary *minority operation* f can be defined by the polymorphism pattern $P = \{((x, x, y), y), ((x, y, x), y), ((y, x, x), x)\}$. In the Boolean domain it is easily verified that the operation defined by this polymorphism pattern is in fact total, but for larger domains the induced operation is properly partial. However, if we remove one of the tuples from P, and for example consider the polymorphism pattern $P \setminus \{((x, y, x), y)\}$, then the class of total operations are known as *Maltsev operations*, and in the Boolean domain we get the partial operation $\phi(0, 0, 0) = 0$, $\phi(0, 0, 1) = 1$, $\phi(1, 1, 0) = 0$, $\phi(1, 0, 0) = 1$, $\phi(0, 1, 1) = 0$, $\phi(1, 1, 1) = 1$. Note that this is precisely the partial operation under consideration in Section 2. We will return to this ternary partial operation several times in the sequel.

A Boolean operation is pSDI if and only if it satisfies a polymorphism pattern. To see this, note that if f is pSDI, then it is easy to create a polymorphism pattern P by letting each tuple $t \in \text{domain}(f)$ such that $f(t) = 0$ correspond to an entry in P. Similarly, it is not difficult to show that any partial operation satisfying a polymorphism pattern must be self-dual and idempotent. Thus, by Theorem 3.13 we first know that if Γ is sign-symmetric and SAT(Γ) is NP-complete then there exists a set of pSDI-operations F such that $[F]_s = \text{pPol}(\Gamma)$, and by the preceding remark every $f \in F$ can be described by a polymorphism pattern. We will now define the pSDI-operations that will play a central role in our current pursuit.

Definition 3.19. We define the following partial operations. Let D be an arbitrary domain.

(1) For $k \geq 3$, the *partial k-ary near-unanimity operation* (partial k-NU operation), denoted nu_k^D, is the k-ary partial operation defined by the pattern which for each $i \in \{1, \ldots, k\}$ contains $((x, x, \ldots, x, y, x, \ldots, x), x)$, where y occurs in the ith position.
(2) For $k \geq 2$, the *partial k-edge operation*, denoted e_k^D, is defined by the polymorphism pattern consisting of $((x, x, y, y, \ldots, y, y), y)$, $((x, y, x, y, \ldots, y, y), y)$, and for each $i \in \{4, \ldots, k{+}1\}$, the tuple $((y, \ldots, y, x, y, \ldots, y), y)$, where x appears in the ith position.

(3) For $k \geq 2$, let A be a $(2^k - 1) \times k$ matrix whose rows enumerate all vectors $\{x, y\}^k \setminus \{x, \ldots, x\}$. Let P be the polymorphism pattern consisting of tuples $(\mathbf{t}, x)$ for every column $\mathbf{t}$ in A. The *k-universal operation*, denoted u_k^D, is the partial operation defined by P.

In all cases, the domain D may be omitted if understood from context; by default, we use $D = \{0, 1\}$.

The operation e_2 is simply the partial Maltsev operation from Section 2 with the first and second argument permuted, and nu_3 is the *partial majority operation*. Note that e_2 and u_2 are equivalent, and that nu_3 is total over the Boolean domain, although it is properly partial over every larger domain. Finally, we note an equivalent definition of u_k. Say that the ith argument of a k-ary partial operation f is *redundant* if there exists $j \neq i$ such that $t[i] = t[j]$ for every $t \in \mathrm{domain}(f)$. Then $\mathrm{u}_k^{\{0,1\}}$ can be defined as the pSDI-operation of arity $2^k - 1$ defined on $2k + 2$ tuples which is not a partial projection and has no redundant arguments. It can be verified that this operation is unique up to permutation of its arguments.

Last, we remark that there is a connection between our notion of polymorphism patterns and the operations studied in connection to the CSP dichotomy (see e.g. the survey by Barto et al. [5]). In technical terms polymorphism patterns essentially matches *strong Maltsev conditions* where the right-hand side is restricted to a single variable. Similar restrictions, called *height-1 identities*, have been considered earlier and it is known that the complexity of a CSP(Γ) problem only depends on the height-1 identities satisfied by the operations in Pol(Γ) [6]. In this context, it may be interesting to note that the reduction used by Cygan et al. [19] showing that Hitting Set is SETH-hard has strong similarities to the more general implementations used in *pp-implementations* and *pp-constructions*, which are associated with the above-mentioned identities [5].

4 THE STRUCTURE OF SIGN-SYMMETRIC CONSTRAINT LANGUAGES

Each pSDI-operation f defined in Definition 3.19 gives rise to a problem Inv(f)-SAT where each relation is invariant under f. For example, in Section 2 we considered the problem Inv(ϕ)-SAT for the partial Maltsev operation ϕ, which is the same problem as Inv(e_2)-SAT. We have already hinted the existence of an improved algorithm for this problem, but before we turn to this matter we determine how the pSDI-operations that we have defined are related to each other. As we will see, these operations, and in fact *all* pSDI-operations, can be precisely classified into "levels" where the strongest operation is the partial k-NU operation, and the weakest operation is the partial k-universal operation. Due to the connection between pSDI-operations and sign-symmetric languages this also gives a precise description of the sign-symmetric languages of interest in the context of proving improved upper bounds.

As a preview of this structure, and of some of the included problems, we refer to Figure 1. The problem and language inclusions illustrated in this figure will be shown across the next two subsections. To simplify this classification we are mainly interested in pSDI-operations that are as "weak" as possible, in the following sense.

Definition 4.1. Let f be a pSDI-operation. We say that f is *trivial* if it is a subfunction of a projection, and a *minimal non-trivial* pSDI-operation if f is non-trivial but every proper subfunction f' of f which is a pSDI-operation is trivial.

It is important to observe that *every* sign-symmetric language Inv(f), is included in a sign-symmetric language Inv(g) where g is a minimal non-trivial pSDI-operation, since Inv(f) $\subseteq$ Inv(g) whenever f is a subfunction of g.

Example 4.2. Let us consider the partial 2-edge operation e_2. Since this operation is defined by the pattern $P = \{((x, x, y), y), ((x, y, x), y)\}$, any pSDI-operation which is a proper subfunction of e_2 is defined either by the pattern $\{((x, x, y), y)\}$, or by the pattern $\{((x, y, x), y)\}$, and in both cases the partial operation is trivial.

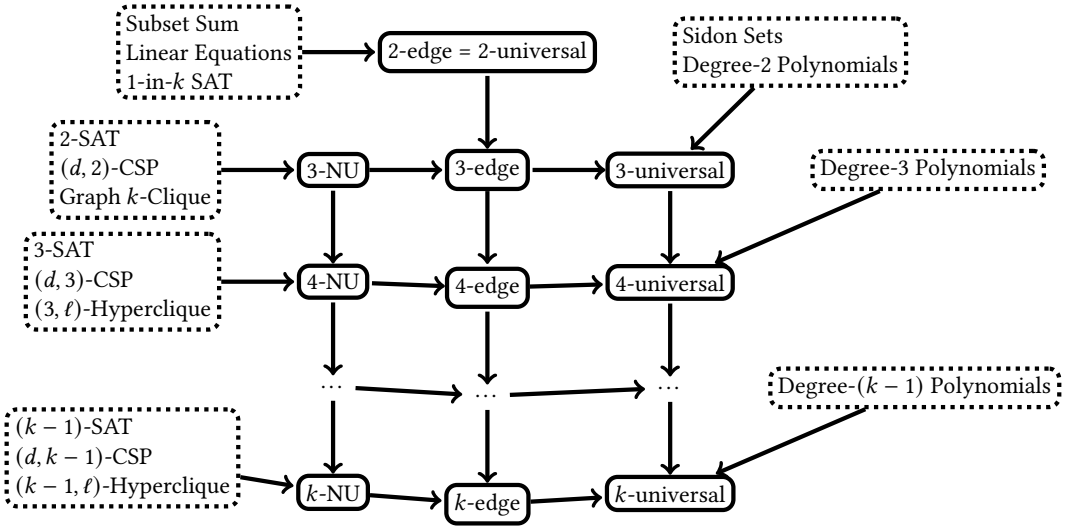

Fig. 1. The inclusion structure between selected minimal pSDI-operations (solid outlines), and some problems that reduce to the corresponding SAT or CSP problem (dotted outlines). Several classes on each level $k \geq 3$ have been omitted.

We invite the reader to verify that the remaining pSDI-operations defined in Definition 3.19 are all minimal. Our study in this section is focused on constraint languages $\Gamma = \mathrm{Inv}(f)$ where f is a single minimal non-trivial pSDI-operation, since these are the most expressive sign-symmetric constraint languages that are still restricted in expressive power. But first, let us make an important note about the power of these languages.

4.1 Sparse and dense languages

We have the following result from Lagerkvist & Wahlström (assuming P ≠ NP).

Theorem 4.3. *[41] Let F be a finite set of partial operations such that* $\mathrm{Inv}(F)$*-SAT is NP-complete. Then any n-ary Boolean relation has a pp-definition over* $\mathrm{Inv}(F)$ *using at most $O(n)$ existentially quantified variables.*

By a counting argument, Theorem 4.3 implies that for any finite set F as in the theorem, the language $\mathrm{Inv}(F)$ can qfpp-define $2^{2^{\Omega(n)}}$ distinct n-ary relations (e.g., there is a double-exponential number of distinct solution spaces for n-variable instances of $\mathrm{Inv}(F)$-SAT). By contrast, for many natural languages, this number is bounded as $2^{n^{O(1)}}$; call a language *sparse* if this applies. Then certainly every finite language is sparse (since then only $n^{O(1)}$ distinct constraints exist over n variables), but furthermore, so is the language where relations are defined as roots of bounded-degree multivariate polynomials over some fixed finite field, as frequently revisited in this article. This follows from the fact that systems of such polynomials have bounded rank; hence any qfpp-definition over this language only needs to use $n^{O(1)}$ polynomials. Since there are at most $2^{n^{O(1)}}$ distinct polynomials once the degree and the (finite) field have been fixed, the sparseness follows (see Jansen & Pieterse [34] and Lagerkvist & Wahlström [42] for extensions and related sparsification applications). With this in mind, it would seem that a complete and explicit characterisation of relations in $\mathrm{Inv}(f)$ is not likely to be useful. Still, for orientation it is interesting to note where specific important problems $\mathrm{SAT}(\Gamma)$ belong in our hierarchy of classes, as we review next.

For minimal non-trivial pSDI-operations f, a strengthening of Theorem 4.3 is possible: The language $\mathrm{Inv}(f)$ is dense enough that padding an n-ary relation R with $O(n)$ randomly chosen parity-check variables is sufficient to create a relation in $\mathrm{Inv}(F)$ with high probability. This will be exploited in Section 6.3 where we explore SETH based lower bounds. Note that Theorem 4.3 does not directly imply a useful SETH lower bound since the introduction of a linear amount of variables per constraint might result in an uncontrolled number of variables in total.

The above notion of sparseness can be compared with the related notion of languages having *few subpowers*, which is effectively languages which remain sparse even under pp-definitions (i.e., when adding existentially quantified variables). It is known that these are precisely the languages which are preserved by a total k-edge-operation for some k [7], and the problem $\mathrm{CSP}(\Gamma)$ for every such language Γ can be solved in polynomial time via the so-called *few subpowers algorithm* [31].

4.2 Properties of specific sign-symmetric constraint languages

In this section, we provide some illustrative examples of languages included in $\mathrm{Inv}(f)$ for particular pSDI-operations f. Despite the density aspect discussed above, we find that the pSDI-operations defined in Section 3.5 do correspond roughly to natural restrictions on the expressive power of a language Γ. We now illustrate the classes with a few examples. In the process we occasionally refer to the language inclusions illustrated in Figure 1. Proofs of these inclusions is given in Theorem 4.16 in Section 4.3. But first, to facilitate the proofs, let us observe some basic properties of pSDI-operations.

Proposition 4.4. *Let f be a pSDI-operation. Then $|\mathrm{domain}(f)| = 2k + 2$ for some $k \in \mathbb{N}$, and f is fully defined either by the non-constant tuples t such that $f(t) = 0$, or by the non-constant tuples t such that $f(t) = 1$.*

Furthermore, when f is minimal, the value k in this proposition defines the *level* of the operation f, and corresponds to the levels in Figure 1.

Consider also an application $f(t_1, \ldots, t_r) = t$ of a pSDI-operation f to n-ary tuples $t_1, \ldots, t_r$, such that the output t is defined. Then for every $i \in [n]$, the tuple $(t_1[i], \ldots, t_r[i])$ corresponds to (at least) one of the patterns in the polymorphism pattern definition of f. Furthermore, $t \notin \{t_1, \ldots, t_r\}$ if and only if for every $j \in [r]$ there is $i \in [n]$ such that $t[i] \neq t_j[i]$, and it can be verified that for all pSDI-operations defined in Section 3 (nu_k, e_k and u_k), this requires that all patterns from the polymorphism pattern definition of f are represented in the application $f(t_1, \ldots, t_r)$. Lemma 4.14 will show more formally that this holds in general for minimal non-trivial pSDI-operations. With this in mind, let us now begin with a basic example.

Lemma 4.5. *Let $k \geq 3$. For every $(k-1)$-ary relation R, we have $R \in \mathrm{Inv}(\mathrm{nu}_k)$.*

Proof. Let $t_1, \ldots, t_k \in R$ be such that $\mathrm{nu}_k(t_1, \ldots, t_k)$ is defined, and for $i \in [k-1]$ let $t^{(i)} = (t_1[i], \ldots, t_k[i])$. For every $i \in [k-1]$, either $t^{(i)}$ is constant or there is a single index j where $t^{(i)}[j]$ deviates from its other entries. By the pigeonhole principle, there is at least one index $j \in [k]$ such that $t^{(i)}[j]$ does not deviate from the majority for any $i \in [k-1]$. Then we have $\mathrm{nu}_k(t_1, \ldots, t_k) = t_j$. □

We also show a corresponding negative statement. By the inclusions shown in the next section, this will imply that a k-clause is not preserved by any operation at "level k" of the hierarchy in Figure 1.

Lemma 4.6. *Let $R \subset \{0,1\}^k$ be a k-clause, i.e., $|R| = 2^k - 1$, $k \geq 2$. Then R is not preserved by the partial k-universal operation.*

Proof. By sign-symmetry, we assume that $R = \{0,1\}^k \setminus \{0^k\}$. Let $t_1, \ldots, t_k$ be the non-constant tuples in domain(u_k) such that $\mathrm{u}_k(t_i) = 0$ for each $i \in [k]$. Then for each $i \in [2^k - 1]$, the tuple $t^{(i)} = (t_1[i], \ldots, t_k[i])$ defines a tuple of R; thus the application

$$\mathrm{u}_k(t^{(1)}, \ldots, t^{(2^k-1)}) = 0^k$$

is defined and shows that $R \notin \mathrm{Inv}(\mathrm{u}_k)$. □

Next, we consider an illustrative example of a useful relation preserved by the partial 2-edge operation.

Lemma 4.7. *Let $R(x_1, \ldots, x_n) \subseteq \{0,1\}^n$ be defined via a linear equation*

$$\sum_{i=1}^{n} \alpha_i x_i = \beta$$

evaluated over a finite field $\mathbb{F}$. Then $R \in \mathrm{Inv}(e_2)$.

Proof. This is a special case of the notion of a *Maltsev embedding* of R previously investigated by the authors [42]. It is known that a relation with a Maltsev embedding is closed under a family of partial operations, of which e_2 is the simplest. □

A particular example of such relations is the Exact SAT problem. We show that its 1-in-k relations are also not closed under nu_k.

Lemma 4.8. *Let $R_{1/k} = \{(x_1, \ldots, x_k) \in \{0,1\}^k \mid x_1 + \ldots + x_k = 1\}$, and $\Gamma_{XSAT} = \{R^s_{1/k} \mid k \geq 1, s \text{ is a } k\text{-ary sign-pattern}\}$. Then $\Gamma_{XSAT} \subseteq \mathrm{Inv}(e_2)$ but is not preserved by nu_k for any k.*

Proof. The positive direction follows from Lemma 4.7, since $R_{1/k}$ can be phrased as a linear equation over the integers mod p, for $p \geq k+1$. The negative direction is immediate: let $R_{1/k} = \{t_1, \ldots, t_k\}$. Then $\mathrm{nu}_k(t_1, \ldots, t_k)$ is defined and equals 0^k. □

Another example of a problem with the character of linear equations is Subset Sum. Even though an instance of Subset Sum is defined by just a single linear equation rather than as a SAT(Γ) instance, we show in Section 6.2 that the complexity of 2-edge-SAT and Subset Sum are closely connected. As for the class $\mathrm{Inv}(e_k)$ for $k \geq 3$, the inclusions illustrated in Figure 1 imply that this class contains both relations with linear equation extensions and all $(k-1)$-clauses.

We are also able to characterise all symmetric relations in $\mathrm{Inv}(e_2)$. This will be needed later for Theorem 4.21.

Lemma 4.9. *Let $R \subseteq \{0,1\}^n$ be a symmetric relation defined by $R(x_1, \ldots, x_n) \equiv (\sum x_i \in S)$ for some $S \subseteq \{0, \ldots, n\}$. Then R is preserved by e_2 if and only if $S = \{i \in [n] \mid i \equiv q \pmod p\}$ for some $p, q \leq n+1$.*

Proof. Let R and S be as in the statement. We first assume that R is preserved by e_2, and let $a, b \in S$ with $a < b$, if possible (if no such pair exists, then S can be produced via $p = n+1$ and $0 \leq q \leq n+1$, and the claim holds). Write $b = a + d$, and assume $a - d \geq 0$. We will define three tuples $t_1, t_2, t_3 \in R$ so that $e_2(t_1, t_2, t_3)$ is defined and returns a tuple of weight $a - d$, which is sufficient to prove the claim since R is symmetric and thus contains all tuples of weight $a - d$. Hence, define $t_1 = 1^{a-d}1^d1^d0^{n-a-d}$, $t_2 = 1^{a-d}1^d0^d0^{n-a-d}$, and $t_3 = 1^{a-d}0^d1^d0^{n-a-d}$. Note that these tuples have weights $a+d$, a, and a, meaning that they are all included in R. If we then apply e_2 we get $e_2(t_1, t_2, t_3) = 1^{a-d}0^d0^d0^{n-a-d}$ of weight $a - d$, and since we assumed that R is invariant under e_2 it follows that this tuple is included in R. A very similar proof shows $b + d \in S$ assuming $b + d \leq n$. Hence, every non-trivial arithmetic sequence in S is complete.

Now let $a, b \in S$ with $a < b$ be chosen to minimise $b - a$. Let $b = a + d$ and write $S' = \{i \in [n] \mid i = a \pmod d\}$. On the one hand, by the above we have $S' \subseteq S$. On the other hand, for any entry $x \in S \setminus S'$ there is an entry $y \in S'$ with $|x - y| < d$, contradicting the choice of a, b. Thus $S = S'$.

In the other direction, let R and S be as described for some parameters p and q, and let $e_2(t_1, t_2, t_3) = t$ be defined for some tuples $t_1, t_2, t_3 \in R$. Let $X_i \subseteq [n]$ for $i = 1, 2, 3$ be the positions j such that $t_i[j] \neq t[j]$; then $X_2 = X_1 \cup X_3$ by definition of e_2. Let w denote the Hamming weight of a tuple and let $x = w(t_1) - w(t)$ and $y = w(t_3) - w(t)$. Assume $x, y \neq 0$ as otherwise $w(t) \in \{w(t_1), w(t_3)\}$ and $t \in R$. Then p divides both $w(t_2) - w(t_1) = x$ and $w(t_2) - w(t_3) = y$. But then also $w(t) = w(t_1) - y = w(t_1) - kp$ for some integer k, hence $w(t) \in S$ and $t \in R$. □

Finally, we show two examples for the partial k-universal operation u_k. The first is a previously studied class of Lokshtanov et al. [48] (as adapted to the Boolean case in Theorem 3.1). They show that if d is fixed and $\mathbb{F}$ is fixed and finite, then this problem admits an improved algorithm, although this is unknown if $\mathbb{F}$ is infinite.

Definition 4.10. Let P_d denote the set of Boolean relations such that each n-ary $R \in P_d$ is the set of roots of an n-variate polynomial over some field $\mathbb{F}$, where each polynomial has degree at most d.

Lemma 4.11. *Let $R \in P_d$ be an n-ary relation. Then R is preserved by (argument paddings of) u_{d+1}, but not by any other non-trivial pSDI-operation of domain size at most $2d + 4$.*

Proof. For the first direction, let $P(x_1, \ldots, x_n)$ be the polynomial defining R, recall that the arity of u_{d+1} is $r = 2^{d+1} - 1$, and let $t_1, \ldots, t_r \in R$ be such that $\mathrm{u}_{d+1}(t_1, \ldots, t_r) = t'$ is defined. We will show that $t' \in R$. Since the set of relations representable by bounded-degree polynomials is sign-symmetric, we may assume for simplicity that $t' = 1^n$. For $i \in [n]$, let $t^{(i)} := (t_1[i], \ldots, t_r[i])$ be the tuple of values taken by the variable x_i in this application of u_{d+1}. We then note that $t^{(i)}$, $i \in [n]$, can take only $d + 2$ different values, corresponding to the elements $x \in \mathrm{domain}(\mathrm{u}_{d+1})$ such that $\mathrm{u}_{d+1}(x) = 1$. We use the tuples $(t_1, \ldots, t_r)$ to define a new polynomial of degree at most d and with at most $d + 1$ variables, by identifying all pairs of variables x_i and x_j that have the same pattern, i.e., if $t^{(i)} = t^{(j)}$. We also eliminate any variable x_i such that $t^{(i)} = 1^r$ by replacing x_i by the constant 1 in P. Let P' be the resulting polynomial, and let R' be the corresponding relation. If $\mathrm{ar}(R') < d + 1$, then by Lemma 4.5 R' is preserved by nu_{d+1} and thus by u_{d+1} as well (see Theorem 4.16). Otherwise, the tuples $t^{(1)}, \ldots, t^{(n)}$ enumerate all $(d + 1)$-tuples except 1^{d+1} (recall the definition of u_d from Definition 3.19). Thus, we have $P'(x_1, \ldots, x_{d+1}) = 0$ for every set of values $x_1, \ldots, x_{d+1} \in \{0, 1\}$ except possibly $x_1 = \ldots = x_{d+1} = 1$. Hence, we will prove that $P'(1, \ldots, 1) = 0$, too. First, for each $I \subset [d + 1]$ let $\chi_I \in \{0, 1\}^{d+1}$ be the tuple such that $\chi_I[i] = 1$ if and only if $i \in I$. Second, we may without loss of generality assume that P' is multilinear, and since its degree is at most d we may in addition assume that it can be written as a sum $\Sigma_{I \subset [d+1]} \alpha_I \prod_{i \in I} x_i$, where each α_I is a coefficient (possibly equal to 0). Hence, $P'(\chi_I) = \sum_{I' \subseteq I} \alpha_{I'}$ for each $I \subset [d + 1]$, and to prove our claim it is therefore sufficient to show that $\alpha_I = 0$ for each $I \subseteq [d + 1]$. Indeed, $\alpha_\emptyset = 0$ since $0^{d+1} \in R'$; and $\alpha_{\{i\}} = 0$ for every $i \in [d + 1]$ since $P'(\chi_{\{i\}}) = \alpha_{\{i\}} + \alpha_\emptyset = \alpha_{\{i\}} = 0$; and so on, in order of increasing cardinality of I. Hence, P' is the constantly-zero polynomial, implying that $1^{d+1} \in R'$, and since P' is an evaluation of P, we have $t' = 1^n \in R$. We have thus shown that relations defined as roots of polynomials of degree d are preserved by the $(d + 1)$-universal operation.

In the other direction, the same argument will show that for any non-trivial pSDI-operation f with $|\mathrm{domain}(f)| \leq 2d+4$ other than the $(d+1)$-universal operation, it is possible to define a polynomial on $(|\mathrm{domain}(f)|-2)/2$ variables and of degree at most d such that the corresponding relation is not preserved by f. Indeed, let $n = (|\mathrm{domain}(f)| - 2)/2$ and $r = \mathrm{ar}(f)$, and let $t_1, \ldots, t_r \in \{0, 1\}^n$ be tuples such that $t^{(i)} = (t_1[i], \ldots, t_r[i])$ for $i \in [n]$ enumerate the non-constant tuples $x \in \mathrm{domain}(f)$ such that $f(x) = 1$ is defined. Then $f(t_1, \ldots, t_r) = 1^n$ is defined, and $1^n \notin \{t_1, \ldots, t_r\}$ since

otherwise f is a partial projection, and is therefore trivial (by choice of the tuples $t^{(i)}$). If $n \leq d$, then we may simply consider the polynomial $P(x_1, \ldots, x_n) = \prod_{i \in [n]} x_i$, whose corresponding relation R is not preserved by f. Otherwise, let $I \subset [d+1]$ be such that $\chi_I \notin \{t_1, \ldots, t_r\}$; this exists since f is not the $(d+1)$-universal partial operation. Let P' be the $(d+1)$-variate polynomial $\sum_{J \subset [d+1]} \alpha_J \prod_{i \in J} x_i$, with coefficients $\alpha_J = 0$ if $I \not\subseteq J$, and with $\alpha_J = (-1)^{|J|-|I|}$ otherwise. The purpose of each monomial $\alpha_J \prod_{i \in J} x_i$ is as follows. First, if $J = I$ then we simply ensure that the monomial $\alpha_I \prod_{i \in I} x_i = \prod_{i \in I} x_i$ is included, which in turn implies that $P'(\alpha_I) = 1$ since any other monomial is 0 under α_I. Second, for any strict superset $J \subset [d+1]$ of I we have (1) $-\prod_{i \in J} x_i$ if $|J| - |I|$ is odd or (2) $\prod_{i \in J} x_i$ if $|J| - |I|$ is even. With this intuition we see that $P'(\chi_I) = 1$, $P'(\chi_J) = 0$ for every $J \subset [d+1]$ distinct from I, and that $P'(1^{d+1}) = -(-1)^{d+1-|I|}$. Hence, the relation corresponding to P' is not preserved by f. □

Finally, we give one example of a symmetric relation in $\mathrm{Inv}(\mathrm{u}_3)$ that has no obvious connection to roots of polynomials, showing that $\mathrm{Inv}(\mathrm{u}_3)$ is richer than the description in Lemma 4.11. A *Sidon set* is a set $S \subseteq \{0, \ldots, n\}$ in which all sums $i + j$ are distinct, for all $i, j \in S$. In other words the set $\{i + j \mid i, j \in S\}$ has size $\binom{|S|}{2} + |S|$.

Lemma 4.12. *Let $S \subseteq \{0, \ldots, n\}$ be a Sidon set, and define the relation $R(x_1, \ldots, x_n) \subseteq \{0,1\}^n$ as*

$$R(x_1, \ldots, x_n) \equiv \left(\sum_{i=1}^{n} x_i \in S \right).$$

Then R is preserved by u_3.

Proof. Assume that there exists $t_1, \ldots, t_7 \in R$ such that $\mathrm{u}_3(t_1, \ldots, t_7) = t \notin R$. For $i \in [n]$, let $x_i = (t_1[i], \ldots, t_7[i])$ be the tuple of values taken by argument i of R in these tuples. Then the tuples x_i take up to 8 different values, partitioned as two constant tuples and three pairs of complementary tuples. Let X_j for $j = 1, 2, 3$ be the set of arguments $i \in [n]$ such that the tuple x_i belongs to the j:th of these pairs, and let n_j be the difference in Hamming weight compared to t if flipping all values belonging to X_j. Let W be the Hamming weight of t. Then S contains the values $W + n_1$, $W + n_2$, $W + n_1 + n_3$ and $W + n_2 + n_3$, forming two pairs of weights with common difference n_3. Since $n_3 \neq 0$ (otherwise $W = W + n_3 \in S$) we must have $n_1 = n_2$. By symmetry, we have $n_1 = n_2 = n_3$. But then S contains the values $W + n_1$, $W + n_1 + n_2 = W + 2n_1$, and $W + n_1 + n_2 + n_3 = W + 3n_1$, which is a contradiction. Thus $n_j = 0$ for at least one j, hence $W \in S$ and $t \in R$, contradicting the original assumption. □

4.3 Structure of minimal non-trivial pSDI-operations

Recall that the *level* of a minimal non-trivial pSDI operation f is $(|\mathrm{domain}(f)| - 2)/2$. In this section we describe minimal pSDI-operations for each level and prove that inclusion structure in Figure 1 holds. We find no examples of minimal non-trivial pSDI-operations on level 0 or 1, and the only non-trivial example on level 2 is the 2-edge operation. At each level $k \geq 3$ the partial k-NU and k-universal operations are the unique strongest and weakest minimal non-trivial pSDI-operation, respectively, whereas the k-edge operation is intermediate. We also find that the k-universal operations u_k are maximally weak in the sense that any non-trivial pSDI-operation with a domain of size $2k + 2$ can define u_k.

To prove these statements, we begin with the following lemma, which formalises one of the main methods of constructing a $(k+1)$-ary partial operation from a k-ary partial operation. We refer to g as an *argument padding* of f.

Lemma 4.13. *Let f be a k-ary partial operation and let g be a $(k+1)$-ary partial operation such that (1)* $\mathrm{Proj}_{1,\ldots,k}(\mathrm{domain}(g)) = \mathrm{domain}(f)$ *and (2) $f(x_1, \ldots, x_k) = g(x_1, \ldots, x_k, x_{k+1})$ for every $(x_1, \ldots, x_k, x_{k+1}) \in \mathrm{domain}(g)$. Then $g \in [f]_s$.*

Proof. Let f and g be as in the statement, and first construct the $(k+1)$-ary partial operation

$$f'(x_1, \ldots, x_k, x_{k+1}) = f(\pi_1^{k+1}(x_1, \ldots, x_k, x_{k+1}), \ldots, \pi_k^{k+1}(x_1, \ldots, x_k, x_{k+1})).$$

Clearly, $f' \in [f]_s$, since it is a composition of f and the projections $\pi_1^{k+1}, \ldots, \pi_k^{k+1}$, and it is not difficult to see that $\mathrm{Proj}_{1,\ldots,k}(\mathrm{domain}(f')) = \mathrm{domain}(f)$ and that g can be obtained as a subfunction of f'. Since $[f]_s$ is closed under taking subfunctions it follows that $g \in [f]_s$. □

More generally, if $f_1, \ldots, f_m$ are partial operations such that each f_{i+1} is an argument padding of f_i, then we also say that f_m is an argument padding of f_1. Using Lemma 4.13 and Theorem 3.5 we may therefore conclude that $\mathrm{Inv}(f) \subseteq \mathrm{Inv}(g)$ whenever g is an argument padding of f. Hence, one useful strategy when comparing the expressive strength of sign-symmetric languages is to fix a pSDI-operation f and then construct stronger languages $\mathrm{Inv}(g)$ via argument padding. We will now prove that this strategy is particularly fruitful for the partial k-NU operation, nu_k, in the sense that any minimal, non-trivial pSDI-operation on level k is an argument padding of nu_k.

Lemma 4.14. *Let f be a pSDI-operation with $|\mathrm{domain}(f)| = 2k+2$, $k \geq 3$. Then f is a minimal non-trivial operation if and only if f is an argument padding of* nu_k.

Proof. In the one direction, assume that f is a padding of nu_k. It is not hard to verify that every subfunction f' of f which is pSDI is a partial projection, and that f is non-trivial. Thus, f is minimal non-trivial. In the other direction, assume that f is minimal and non-trivial, and let $r = \mathrm{ar}(f)$. Let $t_1, \ldots, t_k$ be the non-constant tuples such that $f(t_1) = 0$ is defined. For each $i \in [k]$, let $j_i \in [r]$ be such that making f undefined on t_i and its complement $\overline{t_i}$ leaves a subfunction of $\pi_{j_i}^r$. It follows that for all $a \in [k]$, $t_a[j_i] \neq 0$ if and only if $a = i$. Then the arguments $j_1, \ldots, j_k$ of f define the partial k-NU operation, and f is a padding of it. □

Our claims about the weakest and strongest operations follow from this.

Lemma 4.15. *The following hold.*

(1) *The unique non-trivial non-total pSDI-operation at level $k < 3$ is the partial 2-edge operation.*
(2) *For any minimal non-trivial pSDI-operation f at level $k \geq 3$, we have* $\mathrm{Inv}(\mathrm{nu}_k) \subseteq \mathrm{Inv}(f) \subseteq \mathrm{Inv}(\mathrm{u}_k)$.
(3) *There are at most 2^{2^k-k-1} distinct minimal non-trivial pSDI-operations at level k.*

Proof. *1.* It is easy to verify that no non-trivial operation is possible on level 1. Let f be a non-trivial pSDI-operation on level 2, and let $t_1, t_2 \in \mathrm{domain}(f)$ be the non-constant tuples such that $f(t_1) = 0$. Consider the options for the pairs $(t_1[i], t_2[i])$ for $i \in [\mathrm{ar}(f)]$. If two distinct positions i, i' give identical pairs, then $t[i] = t[i']$ for every $t \in \mathrm{domain}(f)$ and i and i' are redundant arguments in f, which we may assume does not occur. If $t_1[i] = t_2[i] = 0$ for some $i \in [\mathrm{ar}(f)]$ then f is a partial projection. This leaves three possible arguments, and unless all three exist, f will be a total operation. The remaining case is that $f = e_2$.

2. By Lemma 4.14 f is a padding of nu_k, which provides the first inclusion. For the second, we may assume that f has no redundant arguments, since otherwise f is equivalent to an operation with fewer arguments. But then by design, u_k is a padding of f, and the second inclusion follows.

3. By Lemma 4.14, we can restrict our attention to paddings of nu_k. Since f is a pSDI-operation, it is defined by the values of the k non-constant tuples t in the domain with $f(t) = 0$. Let $t_1, \ldots, t_k$ be those tuples, and for $i \in [\mathrm{ar}(f)]$ let $t^{(i)} = (t_1[i], \ldots, t_k[i])$. As above, we may assume that $t^{(i)} \neq t^{(j)}$ for all distinct $i, j \in [\mathrm{ar}(f)]$. This leaves at most 2^k

possible arguments. Furthermore, $t^{(i)}$ cannot be all-zero unless f is a partial projection, and k arguments are determined by nu_k. This leaves $2^k - k - 1$ arguments, whose presence or absence defines f. □

The inclusion structure between the k-NU, k-edge and k-universal partial operations are now straightforward to prove with these results.

Theorem 4.16. *Let $k \geq 3$. Then the following inclusions hold.*

(1) $\mathrm{Inv}(e_2) \subset \mathrm{Inv}(e_k)$,
(2) $\mathrm{Inv}(\mathrm{nu}_k) \subset \mathrm{Inv}(e_k) \subset \mathrm{Inv}(\mathrm{u}_k)$,
(3) $\mathrm{Inv}(\mathrm{nu}_k) \subset \mathrm{Inv}(\mathrm{nu}_{k+1})$,
(4) $\mathrm{Inv}(e_k) \subset \mathrm{Inv}(e_{k+1})$, *and*
(5) $\mathrm{Inv}(\mathrm{u}_k) \subset \mathrm{Inv}(\mathrm{u}_{k+1})$.

Proof. For the inclusions, the second item follows from Lemma 4.15, and every other inclusion follows from Lemma 4.13. Indeed, it is readily verified that for every $k \geq 3$, e_k is an argument padding of e_{k-1} and nu_{k+1} is an argument padding of nu_k. For the universal operations, let $t_1, \ldots, t_{k+1}$ be the non-constant tuples of $\mathrm{domain}(\mathrm{u}_{k+1})$ such that $\mathrm{u}_{k+1}(t_i) = 0$, $i \in [k+1]$. Then the tuples $t^{(i)} = (t_1[i], \ldots, t_{k+1}[i])$, $i \in [2^{k+1} - 1]$ spell out all $(k+1)$-tuples except 0^{k+1}, without repetition. Consider the subset $I \subset [\mathrm{ar}(\mathrm{u}_{k+1})]$ consisting of indices i such that $t_{k+1}[i] = 0$. Note that $t^{(i)}$ for $i \in I$ enumerates all k-tuples except 0^k, padded with a 0. It follows that $\mathrm{Proj}_I(\mathrm{u}_{k+1}) = \mathrm{domain}(\mathrm{u}_k)$ and that u_{k+1} is an argument padding of u_k. By Lemma 4.13 the inclusion follows.

To show that the inclusions are strict, consider the following: a k-clause is preserved by nu_{k+1} (Lemma 4.5) but not by u_k (Lemma 4.6); a 1-in-k constraint is preserved by e_k for every $k \geq 2$ but not by nu_k (Lemma 4.8); and the language P_{k-1} of roots of polynomials of degree at most $k-1$ is preserved by u_k but not by any other operation on level k by Lemma 4.11. □

Finally, we have an easy consequence in more general terms.

Corollary 4.17. *Let f be a pSDI-operation with $|\mathrm{domain}(f)| = 2k + 2$. Then $\mathrm{Inv}(f) \subseteq \mathrm{Inv}(\mathrm{u}_k)$.*

Proof. Let f' be an arbitrary minimal pSDI-operation that is a subfunction of f. Then f' belongs to some level $k' \leq k$, hence $\mathrm{Inv}(f) \subseteq \mathrm{Inv}(\mathrm{u}_{k'}) \subseteq \mathrm{Inv}(\mathrm{u}_k)$ by Lemma 4.15 and Theorem 4.16. □

4.4 Complementary consequences

We now consider some dual questions, i.e., what consequences can we (in general) draw from the information that some sign-symmetric language Γ is not preserved by f, for some pSDI-operation f? We begin with an easy result, which forms the building block of later results.

Lemma 4.18. *Let Γ be a sign-symmetric language which is not preserved by nu_k, for some $k \geq 3$. Then Γ can qfpp-define a k-ary symmetric relation R_k which does not contain any tuples of weight 0 but all tuples of weight 1.*

Proof. Let $k \geq 3$ be an arbitrary constant, and let $R \in \Gamma$ be a relation not preserved by nu_k of some arity $n = \mathrm{ar}(R)$. Let $t_1, \ldots, t_k \in R$ be witnesses to this, i.e., $\mathrm{nu}_k(t_1, \ldots, t_k) = t$ is defined and $t \notin R$. Define $t^{(i)} = (t_1[i], \ldots, t_k[i])$.

By sign-symmetry, we may assume that $t = 0^n$. Furthermore, if there is an argument $i \in [n]$ such that $t^{(i)} = 0^k$, then we can find a smaller counterexample by fixing the ith argument of R to 0. Thus, for every $i \in [n]$, the tuple $t^{(i)}$ now contains precisely one non-zero value. Let us define a new relation $R'(x_1, \ldots, x_k)$ of arity k by identifying arguments

according to this, i.e., for every position $i \in [n]$ such that $t^{(i)}$ is non-zero in position $j \in [k]$, insert the variable x_j in position i in R. Next, define R'' as the result of the conjunction of all $k!$ applications of R' with permuted argument order. Then R'' is a symmetric relation which contains all tuples of weight 1 but none of weight 0. Thus, Γ qfpp-defines a relation $R_k = R''$ as described of every arity $k \geq 3$. □

By a similar strategy, we have an important result about languages not preserved by the k-universal operation.

Lemma 4.19. *Let Γ be a sign-symmetric language not preserved by u_k for some $k \geq 2$. Then Γ can qfpp-define all k-clauses.*

Proof. Let $R \in \Gamma$ be a relation not preserved by u_k, and let $n = \mathrm{ar}(R)$ and $r = 2^k - 1$ be the arity of u_k. Let $t_1, \ldots, t_r \in R$ be such that $\mathrm{u}_k(t_1, \ldots, t_r) = t$ is defined and $t \notin R$. By sign-symmetry of Γ, we may assume that $t = 0^n$. Under this assumption we will prove that Γ can qfpp-define the relation $\{0,1\}^k \setminus \{0^k\}$, i.e., the set of models of the k-clause $(x_1 \vee \ldots \vee x_k)$, which is sufficient since Γ is sign-symmetric. Create a new relation by identifying all variables x_i and x_j in $R(x_1, \ldots, x_n)$ for which $t_a[i] = t_a[j]$ for every $a \in [r]$. Also assume that there is no variable x_i such that $t_a[i] = 0$ for every $a \in [r]$, or else replace x_i by the constant 0 in R (again by sign-symmetry). If we for each $1 \leq i \leq n$ let $t^{(i)} = (t_1[i], \ldots, t_r[i])$, then it is important to note that the sequence $t^{(1)}, \ldots, t^{(n)}$ enumerates all non-constant elements in $\mathrm{domain}(\mathrm{u}_k)$ where u_k returns 0, since otherwise $t_i = 0^n$ for some $i \in [n]$. Hence, the above variable identifications result in a new relation R' of arity precisely k, and it is readily verified that the only k-tuple not included in R' is 0^k. □

By utilising this expressive power, we show a similar (although perhaps less immediately enlightening) consequence for languages not preserved by e_k.

Lemma 4.20. *Let Γ be a sign-symmetric language not preserved by e_k for some $k \geq 2$. Then Γ can qfpp-define the relation R_k defined by*

$$R_k = \{(x_1, \ldots, x_k) \in \{0,1\}^k \mid \sum_{i=1}^{k} x_i = 1\} \cup \{(1, 1, 0, \ldots, 0)\},$$

where the last tuple contains 1 in the first two positions and 0 in all remaining positions.

Proof. We first implement a k-ary relation R' as in Lemmas 4.18 and 4.19. Let $R \in \Gamma$ be a relation not preserved by e_k, of arity $\mathrm{ar}(R) = n$, and let $t_1, \ldots, t_{k+1} \in R$ be tuples such that $e_k(t_1, \ldots, t_{k+1}) = t$ is defined and $t \notin R$. By sign-symmetry, we can assume that $t = 0^n$, and that no tuple $(t_1[i], \ldots, t_{k+1}[i])$ is constantly 0. Thus, there are precisely k different tuples occurring among $(t_1[i], \ldots, t_{k+1}[i])$ for $i \in [n]$. We implement the relation R' by inserting variables x_1 through x_k in positions $i \in [n]$ according to these patterns. We deduce (up to argument ordering) that $R_k \subseteq R'$ and $0^k \notin R'$. . Finally, since Γ is not preserved by e_2, by Lemma 4.19 Γ qfpp-defines all 2-clauses. It can be easily verified that by appropriate usage of constraints $(\neg x_i \vee \neg x_j)$ for $i, j \in [k]$ we can eliminate all tuples of $R' \setminus R_k$ without eliminating any tuple of R_k; hence the result is a qfpp-definition of R_k. □

By the above (Lemmas 4.20 and 4.19) we have simple, 'canonical' consequences of a language not being preserved by e_k respectively u_k for some $k \geq 2$, in that these lemmas provide a single relation R each, such that Γ can qfpp-define R if and only if R is not preserved by the respective partial operation. For the partial k-NU operation, the consequence is more open, but we can actually strengthen this significantly if we assume a constraint language not preserved by *any* partial k-NU operation.

Theorem 4.21. *Let Γ be a sign-symmetric language that is not preserved by the partial k-NU operation, for any k. Then one of the following holds.*

(1) *Γ can qfpp-define 1-in-k-clauses for every k.*
(2) *There is a fixed prime p such that Γ can qfpp-define relations*

$$\sum_{i=1}^{k} x_i \equiv a \pmod{p}$$

for every $0 \leq a < p$, of every arity k.

Before we proceed with the proof, let us make a simple observation about qfpp-definitions among symmetric relations.

Lemma 4.22. *Let R be a symmetric n-ary relation, including tuples of weights $S \subseteq \{0, \ldots, n\}$. Using R, we can qfpp-define symmetric relations of the following descriptions.*

(1) *Shift down: a relation of arity $n-1$ accepting values $S' = \{x - 1 \mid x \in S, x > 0\}$.*
(2) *Truncate: a relation of arity $n-1$ accepting values $S' = \{x \in S \mid x < n\}$.*
(3) *Grouping: for any integer $p > 1$, a relation of arity $\lfloor n/p \rfloor$ accepting values $S' = \{x' \mid x'p \in S\}$.*

Proof. These are implemented by, respectively, fixing an argument to 1 in R; fixing an argument to 0 in R; and grouping arguments of R in groups of size p (after truncating $\mathrm{ar}(R)$ to an even multiple of p). □

We can now show the result. The proof is split into two cases, but let us first consider the case when Γ is preserved by the partial Maltsev operation.

Lemma 4.23. *Let Γ be a sign-symmetric language that is not preserved by the partial k-NU operation for any k, nor by e_2. Then Γ can qfpp-define 1-in-k-clauses for every k.*

Proof. The proof is similar to Lemma 4.20. Since Γ is not preserved by e_2, by Lemma 4.19 Γ qfpp-defines 2-clauses, and by Lemma 4.18 Γ qfpp-defines some symmetric relation R of every arity k which accepts tuples of weight 1 but not tuples of weight 0. Adding all negative 2-clauses to R qfpp-defines the relation $R_{1/k}$ which accepts only tuples of weight 1. □

We now finish the proof.

Proof of Theorem 4.21. If Γ is not preserved by e_2, then the result follows from Lemma 4.23, so assume that e_2 is a partial polymorphism of Γ. By Lemma 4.18, Γ can qfpp-define a symmetric relation R of any arity k which accepts tuples of weight 1 but not tuples of weight 0. By Lemma 4.9 this relation must be equivalent to $\sum X = 1 \pmod{r}$ for some period r (possibly $r > k$). We first note that if Γ can qfpp-define a symmetric relation of arity $r \geq 2k$ which accepts tuples of only one weight w where $w \notin \{0, r\}$, then we can qfpp-define a 1-in-k-clause. Let R' be such a relation. We assume $w \leq r/2$, or else we negate every position of R'. Then we fix $w-1$ arguments of R' to 1 and $r-(w-1+k) \geq 0$ positions to 0, leaving a 1-in-k-clause on the non-fixed positions. Similarly, if Γ can qfpp-define symmetric relations of arbitrarily large period r accepting at least one non-constant tuple, then we can implement a relation R' as above and produce a 1-in-k-clause. Thus, in the remaining case, there is a global bound p such that every symmetric relation qfpp-defined by Γ has a period of at most p. In particular, there is some period $p' \leq p$ that can be implemented for infinitely many arities. If p' is not prime, let $p' = p''p'''$ where p'' is a prime factor of p'; we may simply group variables in groups of p''' variables at a time, and possibly pad with less than p''' zeroes, to implement symmetric relations of period p''. It is also easy to see that we can similarly implement all offsets a by inserting constants. This finishes the proof. □

Section summary. In summary of this section, towards the purpose of discussing sign-symmetric languages Γ such that SAT(Γ) does, or does not, admit an improved algorithm under SETH, we conclude the following. Recall that Γ^k_{SAT} denotes the language of all k-clauses. We find that Γ^k_{SAT} is preserved by every minimal operation on level $k' > k$ (in particular, by nu_{k+1}); not preserved by any operation on a level $k' \leq k$; and that any sign-symmetric language Γ which is not preserved by the k-universal partial operation u_k can qfpp-define Γ^k_{SAT}. Assuming SETH, the minimal non-trivial pSDI-operations that preserve Γ therefore appear to be reasonable proxies for the complexity of SAT(Γ).

Finally, for each level k, there is a language – namely the language of roots of polynomials of degree less than k – which is preserved by u_k but not by any other operation at level $k' \leq k$, and which does admit an improved algorithm [48]. This shows that any "dichotomy" characterising sign-symmetric languages Γ for which SAT(Γ) admits an improved algorithm under SETH, cannot require a minimal non-trivial pSDI-operation other than u_k for some k.

It remains to show that these very mild restrictions, of requiring only the presence of a single non-trivial pSDI-operation f preserving Γ, can be powerful enough to ensure that SAT(Γ) admits an improved algorithm. This is our topic of study for the next section.

5 UPPER BOUNDS FOR SIGN-SYMMETRIC SATISFIABILITY PROBLEMS

In this section, we consider the feasibility of designing an improved algorithm directly for Inv(f)-SAT for a minimal non-trivial pSDI-operation f, i.e., an improved algorithm that only uses the abstract properties guaranteed by such an operation f. Some of the algorithms that we consider are also valid for arbitrary finite domains, and in those cases we by Inv(f)-CSP mean the corresponding operation f defined over the domain in question. We show improved algorithms unconditionally for $f = e_2$ and for $f = \text{nu}_3$, over arbitrary finite domains (where the latter result is only interesting for the non-Boolean case, since the Boolean case is in P). The algorithms for these cases use, respectively, a Subset Sum-style meet-in-the-middle algorithm and fast matrix multiplication over exponentially large matrices. These algorithms all work in the extension oracle model.

We also show conditional or partial results. We show two conditional results for partial k-NU operations, showing that k-NU-CSP admits an improved algorithm in the oracle model if the $(k, k-1)$-hyperclique problem admits an improved algorithm, and that k-NU-SAT admits an improved algorithm in the explicit representation model if the Erdős-Rado *sunflower conjecture* [22] holds for sunflowers with k sets. The first of these results is a direct generalisation of the matrix multiplication strategy; the second uses fast local search in the style of Schöning [59]. Finally, we also consider the symmetric special case of 3-edge-SAT, and show that this problem reduces to a problem of finding a unit-coloured triangle in an edge-coloured graph. This, in turn, follows from fast algorithms for sparse triangle detection. Several of the algorithms we reduce to have a running time that depends on the matrix multiplication exponent ω; the best currently known value is $\omega < 2.373$ [44, 63].

Before we begin, we need the following lemma, which shows that if a relation is preserved by a pSDI-operation, then it is possible to view the relation as a relation of smaller arity over a larger domain, which is preserved by the corresponding partial operation over the larger domain.

Lemma 5.1. *Let R be an n-ary relation over a set of values D, f a pSDI-operation preserving R, and P the polymorphism pattern that defines f. Let $I_1 \ldots, I_m$ be a partition of $[n]$, and $R_{I_1,\ldots,I_m}$ the m-ary relation*

$$R_{I_1,\ldots,I_m} = \{(\text{Proj}_{I_1}(t), \ldots, \text{Proj}_{I_m}(t)) \mid t \in R\}$$

over the set of values $\text{Proj}_{I_1}(R) \cup \ldots \cup \text{Proj}_{I_m}(R)$. *Then the partial operation* f' *defined by* P *over* $\text{Proj}_{I_1}(R) \cup \ldots \cup \text{Proj}_{I_m}(R)$ *preserves* $R_{I_1,\ldots,I_m}$

Proof. Let $k = \text{ar}(f') = \text{ar}(f)$. Let $t_1, \ldots, t_k \in R$ and let $t'_1, \ldots, t'_k \in R_{I_1,\ldots,I_m}$ be the corresponding tuples of $R_{I_1,\ldots,I_m}$. Assume that $f'(t'_1, \ldots, t'_k)$ is defined, i.e., $(t'_1[j], \ldots, t'_k[j]) \in \text{domain}(f')$ for each $j \in [k]$. Let $i \in [n]$ and let I_j be the index set such that $i \in I_j$. Since $f'(t'_1[j], \ldots, t'_k[j])$ is defined it must be an instantiation of a tuple $p \in P$. It follows that $(t_1[i], \ldots, t_k[i])$ must be an instantiation of p as well, implying that $f(t_1[i], \ldots, t_k[i])$ is defined. Hence, f' preserves $R_{I_1,\ldots,I_m}$. □

5.1 An $O^*(|D|^{\frac{n}{2}})$ algorithm for 2-edge-CSP

The following result was shown in Section 2.2, essentially using Lemma 5.1 together with the structure of binary 2-edge-relations.

Theorem 5.2. *2-edge-CSP is solvable in* $O^*(|D|^{\frac{n}{2}})$ *time in both the extension oracle model and the explicit representation.*

5.2 An $O^*(|D|^{\frac{\omega n}{3}})$ algorithm for 3-NU-CSP

The algorithm in Section 5.1 used the rectangularity property of binary relations in order to obtain an improved algorithm for 2-edge-CSP. In this section we will devise an $O^*(|D|^{\frac{\omega n}{3}})$ time algorithm for 3-NU-CSP by exploiting a structural property that is valid for all ternary relations preserved by nu_3. Here, $\omega < 2.373$ is the matrix multiplication exponent. We will need the following definition.

Definition 5.3. An n-ary relation R over D is *k-decomposable* if there for every $t \notin R$ exists an index set $I \subseteq [n]$, $|I| \leq k$, such that $\text{Proj}_I(t) \notin \text{Proj}_I(R)$.

In the total case it is known that R is k-decomposable if R is preserved by a total k-ary NU-operation [35]. In general, this is not true for partial NU-operations, but we still obtain the following result.

Lemma 5.4. *Let* R *be a* k*-ary relation preserved by* nu_k. *Then* R *is* $(k-1)$*-decomposable.*

Proof. Let t be a k-ary tuple not included in R. Assume that $\text{Proj}_I(t) \in \text{Proj}_I(R)$ for every index set $I \subseteq [k]$, $|I| < k$. But then there must exist $t_1, \ldots, t_k \in R$ such that each t_i differ from t in at most one position. This furthermore implies that $\text{nu}_k(t_1, \ldots, t_k)$ is defined, and therefore also that $\text{nu}_k(t_1, \ldots, t_k) = t \notin R$. This contradicts the assumption that nu_k preserves R, and we therefore conclude that there must exist an index set $I \subseteq [k]$ of size at most $k-1$, such that $\text{Proj}_I(t) \notin \text{Proj}_I(R)$. □

Here, it might be instructive to note that any Boolean relation which is $(k-1)$-decomposable can be defined as a conjunction of $(k-1)$-clauses. Hence, Lemma 5.4 properly generalises Lemma 4.5. With the help of $(k-1)$-decomposability we are now ready to prove our main result in this section.

Theorem 5.5. *3-NU-CSP is solvable in* $O^*(|D|^{\frac{\omega n}{3}})$ *time in both the extension oracle model and the explicit representation, where* $\omega < 2.373$ *is the matrix multiplication exponent.*

Proof. Let (V, C) be an instance of 3-NU-CSP where $V = \{x_1, \ldots, x_n\}$ and $C = \{C_1, \ldots, C_m\}$. Partition $[n]$ into three sets I_1, I_2, I_3 such that $|I_i| = \frac{n}{3}$ (or, if this is not possible, as close as possible). Let F_1, F_2, F_3 denote the set of all partial truth assignments corresponding to I_1, I_2, I_3, and observe that $|F_i| \leq |D|^{\frac{n}{3}+1}$ for each $1 \leq i \leq 3$. First, for each partial

truth assignment $f \in F_i$, remove it from the set F_i if there exists a constraint in the instance which is not consistent with f. This can be done in polynomial time with respect to the number of constraints in the instance, using an extension oracle query for each constraint. Second, construct a 3-partite graph where the node set is the disjoint union of F_1, F_2 and F_3, and add an edge between two nodes in this graph if and only if the combination of this partial truth assignment is not contradicted by any constraint in the instance. Last, answer yes if and only if the resulting graph contains a triangle.

We begin by proving correctness of this algorithm and then analyse its complexity. We first claim that if $f_1 \in F_1, f_2 \in F_2, f_3 \in F_3$ forms a triangle in the 3-partite graph, then the combination of f_1, f_2, f_3 satisfies each constraint in the instance. Suppose otherwise, i.e., that there exists $R(x_{i_1}, \ldots, x_{i_k}) \in C$, $k = \mathrm{ar}(R)$, which is not satisfied by the combination of f_1, f_2, f_3. Let $I_1' = \{j \mid i_j \in I_1\}$, $I_2' = \{j \mid i_j \in I_2\}$, and $I_3' = \{j \mid i_j \in I_3\}$ and consider the relation $R_{I_1', I_2', I_3'} = \{(\mathrm{Proj}_{I_1}(t), \mathrm{Proj}_{I_2}(t), \mathrm{Proj}_{I_3}(t)) \mid t \in R\}$ over the set of values $\mathrm{Proj}_{I_1}(R) \cup \mathrm{Proj}_{I_2}(R) \cup \mathrm{Proj}_{I_3}(R)$. By Lemma 5.1 this relation is preserved by the nu_3 operation over the larger domain, and it then follows from Lemma 5.4 that this relation is 2-decomposable. But then there exists $y_1, y_2 \in \{f_1, f_2, f_3\}$ whose combination does not satisfy $R(x_{i_1}, \ldots, x_{i_k})$, contradicting the assumption that f_1, f_2, f_3 forms a triangle. Hence, if (V, C) is satisfiable, then there clearly exists a triangle in the 3-partite graph, and if there exists a triangle, then by following the reasoning above, the instance must be satisfiable.

For the complexity, we begin by enumerating the three sets of partial truth assignments, which takes $O(|D|^{\frac{n}{3}})$ time. We then remove any partial truth assignment which is not consistent with the instance, which increases this by a polynomial factor, depending only on the number of constraints and the extension queries for each constraint. Similarly, when constructing the 3-partite graph we enumerate all binary combinations of partial truth assignments from the three sets and check whether they are consistent. After this we check for the existence of a triangle in the resulting graph with $O(|D|^{\frac{n}{3}})$ nodes, which can be solved in $O(|D|^{\frac{n}{3}\omega})$ time for $\omega < 2.373$, using fast matrix multiplication, i.e., by cubing the adjacency matrix and then scanning the diagonal for non-zero entries. □

5.3 Strategies for k-NU-SAT

It is easy to see that the strategy used in Theorem 5.5 extends to reducing k-NU-CSP problems to $(k, k-1)$-HYPERCLIQUE, i.e., the problem of finding a k-vertex hyperclique in a $(k-1)$-regular hypergraph. Thus we get the following.

LEMMA 5.6. *Assume that $(k, k-1)$-HYPERCLIQUE on n vertices can be solved in time $O^*(n^{k-\varepsilon})$ for some $\varepsilon > 0$. Then k-NU-CSP admits an improved algorithm in the extension oracle model, i.e., an algorithm running in time $O^*(|D|^{(1-\varepsilon')n})$ on domain size D and on n variables, for some $\varepsilon' > 0$.*

However, it should be noted that this is a notoriously difficult problem, and there is some evidence against such results [45]. Thus, we also investigate a less general algorithm that rests on a milder assumption.

5.3.1 k-NU-SAT via local search. We show that subject to a popular conjecture, k-NU-SAT admits an improved algorithm in the explicit representation model via a local search strategy. To state this we need a few basic definitions. A *sunflower (with k sets)* is a collection of k sets $S_1, \ldots, S_k$ with common intersection $S = S_1 \cap \ldots \cap S_k$, called the *core*, such that for every pair $i, j \in [k]$, $i \neq j$, we have $S_i \cap S_j = S$. Note that we may have $S = \emptyset$. The *sunflower conjecture* [22], in the form we will need, states that for every k there is a constant C_k such that for every n, every collection of at least C_k^n sets of cardinality n contains a sunflower with k petals. This conjecture was the subject of the Polymath 10 collaborative

mathematics project, but remains a notorious open problem. See Alon, Shpilka and Umans [1] for variations of the conjecture and connections to other problems.

We first show a simple connection between the sunflower conjecture for sunflowers with k sets and relations $R \in \mathrm{Inv}(\mathrm{nu}_k)$. For convenience, for a set $S \subseteq [n]$ we denote by χ^n_S the tuple $t \in \{0,1\}^n$ such that for each $i \in [n]$, $t[i] = 1$ is $i \in S$ and $t[i] = 0$ otherwise.

Lemma 5.7. *Let $R \subset \{0,1\}^n$ be a relation with $0^n \notin R$. Say that a tuple $t = \chi^n_S$ is* minimal *in R if $t \in R$ but for every $S' \subset S$ we have $\chi^n_{S'} \notin R$. For $i \in [n]$, let $\mathcal{F}_i$ be the set of minimal tuples in R of Hamming weight i. If R is preserved by nu_k, then $\mathcal{F}_i$ does not contain a sunflower of k sets.*

Proof. Let $\mathcal{F}_i$ be as in the statement, and assume that R is preserved by nu_k. Assume that there are distinct sets $S_1, \ldots, S_k$ forming a sunflower with some core S, such that $\chi_{S_j} \in \mathcal{F}_i$ for every $j \in [k]$. But then the operation $\mathrm{nu}_k(\chi_{S_1}, \ldots \chi_{S_k})$ is defined, and produces the tuple χ_S. This contradicts that the tuples are minimal in R. □

We show that the sunflower conjecture is sufficient to allow an improved algorithm. The key idea is that if a relation R is given explicitly, the sun flower conjecture implies that all minimal tuples of R of Hamming weight p can be enumerated in $O^*(2^{O(p)})$ time, which makes a local search algorithm possible.

Lemma 5.8. *Assume that the sunflower conjecture holds for sunflowers with k sets, with some constant C_k. Let Γ be a sign-symmetric language preserved by nu_k. Assume that for every n-ary relation $R \in \Gamma$ and every $p \in [n]$, the minimal tuples in R of Hamming weight at most p can be enumerated in time $O^*(2^{O(p)})$. Then SAT(Γ) admits an improved algorithm.*

Proof. We first show that the assumptions are sufficient to allow a solution for the *local search* problem for SAT(Γ), in the following form. Let an instance (V, C) of SAT(Γ) with $|V| = n$, a tuple $t \in \{0,1\}^n$, and an integer $p \in [n]$ be provided. We can in $O^*(2^{O(p)})$ time decide whether there is a tuple $t' \in \{0,1\}^n$ with Hamming distance at most p from t that satisfies (V, C).

For this, we repeatedly perform the following procedure. Verify whether the present tuple t satisfies (V, C), and if not, let $R(X)$ be a constraint in C falsified by t, and let $I \subseteq [n]$ be the set of indices corresponding to the set of variables X. Let s be the sign pattern such that $(\mathrm{Proj}_I(t))^s = 0^{|X|}$. Note that $R^s \in \Gamma$ by assumption. We then enumerate the minimal tuples in R^s of Hamming weight at most p, and for every such tuple t', of weight i, let t'' be the tuple t with bits flipped according to t', and recursively solve the local search problem from tuple t'' with new parameter $p - i$. Correctness is clear, since the search is exhaustive (because we loop through all minimal tuples). We argue that this solves the local search problem itself in $O^*(2^{O(p)})$ time. For the running time, assume for simplicity that producing the tuples takes $O^*(c^p)$ time and, for the same constant c, there are at most c^i minimal tuples of weight i (by Lemma 5.7). Up to polynomial factors, the running time is then bounded by a recurrence

$$T(p) = c^p + \sum_{i=1}^{p} c^i T(p - i),$$

which is bounded as $T(p) \leq (2c)^p$.

From here on, well-known methods can be used to complete the above into an improved algorithm: sample c^n random points, for a suitable c, and perform local search around the p-neighbourhood of each point. For additional details, cf. Schöning's algorithm for k-SAT [59] and its derandomisation [21], or, alternatively, restrict the above to *monotone* local search instead of arbitrary local search and apply the method of Fomin et al. [24]. □

In particular, this is allows for an algorithm in the explicit representation model.

Theorem 5.9. *Assume that the sunflower conjecture holds for sunflowers with k sets. Then k-NU-SAT admits an improved algorithm in the explicit representation model.*

We leave it as an open question whether access to an extension oracle (also known as an interval oracle) suffices to solve the local search problem in single-exponential time. The problem, of course, is that the bounds above only apply to the *minimal* tuples, and while it is easy to find a single minimal tuple using an extension oracle, it is less obvious how to test for the existence of a minimal tuple within a given interval. Meeks [49] showed how a similar result is possible, but her method would require an oracle for finding minimal satisfying tuples of weight *exactly* i, which is also not clear how to do.

5.3.2 k-NU-SAT and bounded block sensitivity. Finally, we briefly investigate connections between the nu_k partial operation and a notion from Boolean function analysis known as *block sensitivity*, introduced by Nisan [51]. See also the book by O'Donnell [52].

We first introduce some temporary notation. For any relation $R \subseteq \{0,1\}^n$, let $f_R : \{0,1\}^n \to \{0,1\}$ be a function defined as $f_R(t) = [t \in R]$, i.e., $f_R(t) = 1$ if $t \in R$ and $f_R(t) = 0$ otherwise. For a tuple $t \in \{0,1\}^n$ and a set $S \subseteq [n]$, let t^S denote the tuple t with the bits of S flipped. A function $f : \{0,1\}^n \to \{0,1\}$ has *block sensitivity* bounded by b if for every $t \in \{0,1\}^n$ there are at most b disjoint sets $S_1, \dots, S_b \subseteq [n]$ such that $f(t^{S_i}) \neq f(t)$ for every $i \in [b]$. We show that nu_k can be seen as a one-sided version of block sensitivity.

Lemma 5.10. *Let $R \subseteq \{0,1\}^n$ be a relation. Then f_R has block sensitivity less than k if and only if both R and its complement $\overline{R} := \{0,1\}^n \setminus R$ are preserved by* nu_k.

Proof. In the first direction, assume that f has block sensitivity at least k. Let $t \in \{0,1\}^n$ be a tuple and let $X_1, \dots, X_k$ be disjoint non-empty subsets of $[n]$ such that for each $1 \leq i \leq k$, we have $f(t^{X_i}) \neq f(t)$. Then if $f(t) = 1$, then the tuples t^{X_i} form a witness that R is not preserved by nu_k, and if $f(t) = 0$ they form a witness against $\overline{R}$ being preserved by nu_k. In the other direction, let $t_1, \dots, t_k \in R$ be such that $\mathrm{nu}_k(t_1, \dots, t_k) = t$ is defined and $t \notin R$. For $i \in [k]$, let X_i be the positions j where $t[j] \neq t_i[j]$. Then $X_1, \dots, X_k$ are disjoint non-empty subsets of $[n]$ showing that f has block sensitivity at least k. The case that $\overline{R}$ is not preserved by nu_k, instead of R, is completely dual. □

It is known that a block sensitivity of at most b implies a *certificate complexity* of at most b^2, i.e., for any relation $R \in \mathrm{Inv}(\mathrm{nu}_k)$ and any tuple $t \in \overline{R}$, there are at most b^2 bits in t that certify that $t \notin R$ [51]. This suggests a branching or local search algorithm for $\mathrm{SAT}(\Gamma)$ where Γ contains such relations. However, more strongly, it implies that R has a decision tree of bounded depth [51], and thus, since k is a constant, that R only depends on constantly many arguments. Thus, block sensitivity is a significantly stronger restriction than what nu_k imposes.

However, one related question remains. Assume that R is an n-ary relation preserved by nu_k, and which does depend on all its arguments. Is there a non-trivial upper bound on $|R|$, e.g., does it hold that $|R| \leq (2 - \varepsilon_k)^n$ for some ε_k depending on k? A positive answer to this question would imply a trivial improved algorithm for k-NU-SAT via enumeration of satisfying assignments, constraint by constraint.

5.4 Symmetric 3-edge-SAT

We finish this section with a result showing that a number of special cases of 3-edge-CSP admits an improved algorithm via sparse triangle finding. The class in particular contains 3-edge-SAT for symmetric relations $R \in \mathrm{Inv}(e_3)$. We begin

by characterising the symmetric relations in $\mathrm{Inv}(e_3)$. In this context, for a relation of arity n, we say that an arithmetic progression $a, a+b, \ldots$ is *complete* if it contains all values $i \in \mathbb{N}$, $0 \leq i \leq n$, such that $i \equiv a \pmod{b}$.

Lemma 5.11. *Let $R \subseteq \{0,1\}^n$ be a symmetric relation preserved by e_3. Let $S \subseteq \{0,\ldots,n\}$ be the weights accepted by R. Then either S is a complete arithmetic progression (possibly a trivial one, of length 1), or $S = \{a, a+b\}$ or $S = \{n-a, n-a-b\}$ for some $a < b$.*

Proof. Let us first make a simpler claim: if $a, a+b \in S$ is a pair that does not extend to a complete progression in S, then either $a - b < 0$ or $a + 2b > n$.

To see this, let $a, a+b \in S$, and assume $a+2b \notin S$, $a+2b \leq n$. First assume $a \geq b$. We subpartition $[n]$ into one set T_0 of size $a - b \geq 0$ and three sets T_i of size b, $i = 1, 2, 3$. This is possible since $a - b + 3b = a + 2b \leq n$. Let $t = \chi_{T_0 \cup \ldots \cup T_3}$ and for $i = 1, 2, 3$ let $t_i = t^{T_i}$. Finally, let $t_4 = t^{T_1 \cup T_2}$. Then the weight of t_i for $i = 1, 2, 3$ is $a + b$ and the weight of t_4 is a, hence $t_1, \ldots, t_4 \in R$; furthermore $e_3(t_4, t_1, t_2, t_3)$ is defined and produces t. Thus we conclude $a < b$, i.e., $a - b < 0$. By the symmetric argument, if $a, a+b \in S$ with $a - b \geq 0$ and $a - b \notin S$, then $a + 2b > n$. This finishes the claim.

Next, assume that $|S| > 2$ and that S contains some pair $a, a+b$ such that the progression does not continue. Let $b > 0$ be the smallest value such that such a pair exists, and again by symmetry assume that $a + 2b \leq n$; thus $a - b < 0$. Let $c \in S \setminus \{a, a+b\}$. First assume $c > a + 2b$. Then we may, similarly to above, pack sets with $|T_0| = a$, $|T_1| = |T_2| = b$, and $|T_3| = c - a - 2b$, and we have a witness showing $a + 2b \in S$. Otherwise, a, $a+b$ and c can form a pair which by choice of b must extend to a complete progression in S. Indeed, if $c < a + b$ then $|a - c| < b$, and otherwise $a + b < c < a + 2b$. Thus S contains a complete progression with period less than b, and consequently there exists a value $c' \in S$ with $a < c' < a + b$. Iterating this step eventually produces an arithmetic progression of step size dividing b, covering a and $a + b$, contradicting the assumption that $a + 2b \notin S$. Thus $|S| = 2$, i.e., $S = \{a, a+b\}$. □

In particular, this lemma shows that every symmetric relation in $\mathrm{Inv}(e_3)$ is a simple arithmetic progression. By Lemma 4.9, every such relation also extends in a simple way to a larger relation in $\mathrm{Inv}(e_2)$ by completing the progression, which we call a *2-edge embedding* of the relation.

We now describe the algorithm. Let R be a relation with arguments X. For a partition $X = X_1 \cup X_2$ and an assignment f to X_1, we refer to the *2-edge label* of f as the pair (f_0, g_0) produced by first extending f to a lex-min assignment g_0 such that $(f, g_0) \in R$, then extending g_0 to a lex-min assignment f_0 such that $(f_0, g_0) \in R$. Note that this is the same procedure used in the algorithm for 2-edge-CSP.

We extend this to 3-partite graphs as follows. Let the variable set be partitioned as $[n] = X \cup Y \cup Z$, and define a graph $G = (V, E)$ with partition $V = V_X \cup V_Y \cup V_Z$, where the nodes of each part represent partial assignments as in Section 5.2. For each edge, verify that the corresponding partial assignment is consistent with each relation in the input instance. We proceed to give labels to edges of G for each relation R as follows. We assume that for each relation, the "type" of R is known to us (2-edge, 3-NU, or symmetric 3-edge). If $R \in \mathrm{Inv}(\mathrm{nu}_3)$, all edges get the same label. Otherwise, let $\hat{R} \supseteq R$ be the 2-edge-embedding of R (with $\hat{R} = R$ if R is already 2-edge). Let pq be an edge in G, corresponding to partial assignments p, q. If one of these assignments, say p, is an assignment to X, then we set the label of pq to the 2-edge label of p in the partition $X \cup (Y \cup Z)$. Otherwise, $p \cup q$ is an assignment to $Y \cup Z$, and we set the label of pq to the 2-edge label of this assignment in $X \cup (Y \cup Z)$. We show that this label scheme captures our language.

Lemma 5.12. *Let R be a relation with arguments U, for some $U \subseteq [n]$, and let $G = (V, E)$ and $X \cup Y \cup Z$ be as above. If either $R \in \mathrm{Inv}(e_2)$, or $R \in \mathrm{Inv}(\mathrm{nu}_3)$, or R is Boolean, symmetric and $R \in \mathrm{Inv}(e_3)$, then a triple (f, g, h) with $f \in V_X$, $g \in V_Y$, $h \in V_Z$ satisfies R if and only if fgh is a triangle in G where the edges fg, fh, gh all have the same label.*

Proof. Refer to a triangle fgh with all edge labels identical as a *single-label triangle*. We will also slightly abuse notation by treating R as a 3-ary relation taking values from $V_X \times V_Y \times V_Z$. First assume that $R \in \mathrm{Inv}(e_2)$, and recall that R is rectangular. Let fgh be a single-label triangle with shared label $L = (f_0, g_0h_0)$; we show that $(f, g, h) \in R$. Since L is the label of the edge gh, it must be that $(f_0, g, h), (f_0, g_0, h_0) \in R$, and by the edges fg and fh it must be that $(f, g_0, h_0) \in R$ as well. By the partial 2-edge operation, this implies $(f, g, h) \in R$. Thus every single-label triangle corresponds to a satisfying assignment.

In the other direction, let $(f, g, h) \in R$. Since R is rectangular, there is a unique lex-min pair (f_0, g_0h_0) in the biclique containing (f, gh), and both extensions (f, g_0h_0) and (f_0, gh) are compatible with R. Thus all three edges get the same label and the algorithm works for $R \in \mathrm{Inv}(e_2)$.

The case $R \in \mathrm{Inv}(\mathrm{nu}_3)$ is trivial. Since such a relation is 2-decomposable, the entire verification of R happens in the stage where edges are filtered, and in the remaining graph, every triangle represents a satisfying assignment and every triangle is single-label.

Finally, assume $R \in \mathrm{Inv}(e_3)$ and is symmetric. If $R \in \mathrm{Inv}(e_2)$, then we argue as above. Otherwise, by Lemma 5.11, either $S = \{a, a + b\}$ or $S = \{n - a, n - a - b\}$ for $a < b$, and $\hat{R}$ verifies that each assignment (f, g, h) has the correct weight when computed mod b. First assume that fgh is a single-label triangle in G and that $S = \{a, a + b\}$. By the edge-filtering step, we know that for each of the edges fg, gh, fh the corresponding partial assignment has weight at most $a + b$. Thus the total weight of (f, g, h) is at most $(a + b)(3/2) \leq a + b + (a + b)/2 < a + 2b$. Dually, assume $S = \{n - a - b, n - a\}$. No edge in fgh has more than $a + b$ zeroes, thus the total assignment has weight greater than $n - a - 2b$. In both cases, since the edge-labels work to verify the value mod b, we conclude $(f, g, h) \in R$.

On the other hand, assume $(f, g, h) \in R$. Since the edge labels are minimal for the more permissive relation $\hat{R}$, the triangle fgh is a single-label triangle. □

The remaining problem can now be solved via algorithms for triangle-finding in sparse graphs.

Theorem 5.13. *Assume a CSP or SAT problem with the following characteristic: for every relation R, either $R \in \mathrm{Inv}(e_2)$ and R is labelled with type e_2, or $R \in \mathrm{Inv}(\mathrm{nu}_3)$ and R is labelled with type nu_3, or the language is Boolean, R is a symmetric relation in $\mathrm{Inv}(e_3)$ and R is labelled with type e_3. This problem can be solved in time $O^*(|D|^{\frac{\omega+3}{6}})$ in the extension oracle model, where $\omega < 2.373$ is the matrix multiplication exponent.*

Proof. By the description above, we create a 3-partite graph G on $3|D|^{n/3}$ vertices (where $|D| = 2$ in the Boolean case), and for every edge in G we give it a vector of labels, one label per relation in the input instance. We refer to this vector as the *colour* of the edge. Note that a symmetric relation R can be "inspected" using its extension oracle to find out the set S of accepted weights. By Lemma 5.12, the instance has a satisfying assignment if and only if G has a triangle where all edges have the same colour.

This we solve as follows. For every colour c used by an edge in G, we generate the graph G_c consisting of all edges of colour c. Let m_c be the number of edges of G_c, and let $N \leq 3|D|^{n/3}$ be the number of vertices in G. We check if G_c contains a triangle. If G_c is dense enough, then we use the usual triangle-finding algorithm for this, with running time $O^*(N^\omega)$, otherwise we use an algorithm for triangle finding in sparse graphs. Alon, Yuster and Zwick [2] show such an algorithm with running time $O(m_c^{2\omega/(\omega+1)})$, where $\omega < 2.373$ is the matrix multiplication exponent. Hence, the crossover point at which we use the dense algorithm is $m_c \geq N^{(\omega+1)/2} =: N^\alpha$. Summing over all colours, we have $\sum_c m_c \leq N^2$. Since the algorithm for sparse graphs has a super-linear running time, the worst case is when we are at the crossover density and use the sparse algorithm $N^{2-\alpha}$ times for a cost of $O(N^\omega)$ each time. This works out to a total

running time $O(N^{(\omega+3)/2})$ for triangle-finding, i.e., the CSP is solved in time $O^*(|D|^{(\omega+3)n/6}) = O^*(|D|^{0.896n})$ using $\omega = 2.373$. □

It is an interesting question whether this strategy can be modified to work for arbitrary relations $R \in \text{Inv}(e_3)$.

Section summary. We have proved that it is indeed feasible to construct improved algorithms for $\text{Inv}(f)$-SAT and $\text{Inv}(f)$-CSP for individual pSDI-operations f. A crucial step for constructing algorithms of this form is first to identify non-trivial properties of relations invariant under f, which for the partial 2-edge operation turned out be rectangularity, and for the partial 3-NU operation 2-decomposability. However, it might not always be the case that every invariant relation satisfies such a clear-cut property, and for 3-edge-SAT we had to settle for an improved algorithm for symmetric relations.

For k-NU-CSP and k-NU-SAT we also gave conditional improvements in terms of $(k, k-1)$-HYPERCLIQUE and the sunflower conjecture.

6 LOWER BOUNDS

In this section we turn to the problem of proving lower bounds for sign-symmetric SAT problems.

6.1 Lower bounds based on k-SAT

As an easy warm-up, we first consider languages Γ such that $\text{SAT}(\Gamma)$ is at least as hard as k-SAT for some k. Recall that c_k for $k \geq 3$ denotes the infimum of the set $\{c \mid k\text{-SAT is solvable in } O(c^n) \text{ time}\}$. Under the ETH, $c_k > 1$ for each $k \geq 3$, and for each $k \geq 3$ there exists $k' > k$ such that $c_{k'} > c_k$ [32]. The best known upper bounds yield $c_k \leq 2 - \Theta(1/k)$, but no methods for lower-bounding the values c_k are known.

Recall that Lemma 4.19 gives a condition under which a language Γ can qfpp-define all k-clauses. We observe the immediate consequence of this.

LEMMA 6.1. *Let Γ be a sign-symmetric constraint language not preserved by the k-universal partial operation. Then $\text{SAT}(\Gamma)$ cannot be solved in time $O^*(c^n)$ for any $c < c_k$, even in the non-uniform model.*

PROOF. By Lemma 4.19, Γ can qfpp-define all k-clauses. More concretely, there is a finite set $\Gamma' \subseteq \Gamma$ of relations such that every k-clause has a fixed, finite-sized gadget implementation over Γ' without using additional variables. Thus, given a k-SAT instance on n variables, we can produce an equivalent instance of $\text{SAT}(\Gamma')$ in linear time, with the same variable set. □

As a consequence, c_k is also a lower bound on the running time for $\text{Inv}(f)$-SAT for every minimal pSDI-operation at level $k+1$ and higher. However, this above lemma applies to any sign-symmetric constraint language, and not just to the special case when $\Gamma = \text{Inv}(f)$. We can also observe a similar consequence for SETH-hardness.

COROLLARY 6.2. *Let Γ be a sign-symmetric constraint language not preserved by the k-universal partial operation for any k. Then assuming SETH, $\text{SAT}(\Gamma)$ does not admit an improved algorithm, even in the non-uniform model.*

PROOF. By SETH, there is for every $\varepsilon > 0$ a constant k such that k-SAT cannot be solved in $O^*((2-\varepsilon)^n)$ time. By Lemma 6.1, there is a reduction from k-SAT to $\text{SAT}(\Gamma)$ for this k. Thus, $\text{SAT}(\Gamma)$ does not admit an improved non-uniform algorithm. □

6.2 2-edge-SAT and Subset Sum

Next, we sharpen the connection between Subset Sum and 2-edge-SAT. Recall that an instance of Subset Sum consists of a set $S = \{x_1, \ldots, x_n\}$ of n numbers and a target integer t, with the question of whether there is a set $X' \subseteq S$ such that $\sum X' = t$. This can also be phrased as asking for $z_1, \ldots, z_n \in \{0, 1\}$ such that

$$\sum_{i=1}^{n} z_i x_i = t.$$

Also recall from Lemma 4.7 that such a relation is contained in $\mathrm{Inv}(e_2)$. However, this does not by itself imply a problem reduction, since an instance of 2-edge-SAT assumes the existence of an extension oracle for every constraint. We show that such a reduction can be implemented by splitting the above equation apart into several equations, based on the bit-expansion of t.

Theorem 6.3. *If 2-edge-SAT is solvable in $O(c^n)$ time for $c > 1$ in the extension oracle model, then Subset Sum is solvable in $O(c^{n+o(n)})$ time.*

Proof. Let $x_1, \ldots, x_n, t \in \mathbb{N}$ be the input to a Subset Sum instance. We will reduce this instance in subexponential time to a disjunction over 2-edge-SAT instances on n variables each.

We proceed as follows. We first apply a procedure of Harnik and Naor [27] that reduces a Subset Sum instance to one of bit length at most $2n + \log \ell$, where ℓ is the bit length of the input. If $\ell \geq 2^n$, then we solve the instance by brute force in time polynomial in the input length, otherwise we are left with an instance of bit length $\ell' \leq 3n$.

Next, set a parameter $k = \sqrt{n}$, and split the binary expansion of the input integers into k blocks of equal length, giving $\sqrt{n}$ blocks of some length $d = O(\sqrt{n})$. For each $i \in [n]$, and each $0 \leq b < k$, let $x_{i,b}$ consist of the bits of the b:th block in the binary expansion of x_i, i.e., $x_i = \sum_{b=0}^{k-1} 2^{di} x_{b,i}$ and $0 \leq x_{i,b} < 2^d$ for every i and b. For each block guess the contribution of the solution to the target value; i.e., if $I \subseteq [n]$ represents the solution so that $\sum_{i \in I} x_i = t$, then for each block b we guess the value $t_b = \sum_{i \in I} x_{i,b}$. Note that $t_b < n2^d$, hence the largest overflow that can carry over to the next block is $t_b / 2^d < n$. It follows that for a single block there are $O(n^2)$ options for t_b, computed from t by considering the incoming and outgoing overflow values. We get at most $O(n^k) = 2^{o(n)}$ guesses in total by guessing all overflows, after which we have replaced the original equation $\sum_i z_i x_i = t$ by the conjunction of $\sqrt{n}$ linear equations, each with a target integer of $O(\sqrt{n})$ bits. This allows us to implement an extension oracle for every such constraint with a running time of $2^{O(\sqrt{n})}$, using a tabulation approach (cf. the well-known dynamic programming algorithm for the knapsack problem).

This encodes an instance of 2-edge-SAT in the extension oracle model with n variables. Using an algorithm for this problem, and multiplying its running time by the time required for answering an oracle query, yields the claimed running time for Subset Sum. □

Given that the running time for 2-edge-SAT in the extension oracle model given in this article matches the best known running time for Subset Sum, and given that improving the latter is a long-open problem, it seems at the very least that an improvement to 2-edge-SAT would require significant new ideas.

6.3 Padding formulas

We now give a combinatorial interlude, showing how relations $R \subseteq \{0, 1\}^n$ can be padded with additional variables such that the new relation lies in $\mathrm{Inv}(f)$, for any non-total partial operation f. This will be leveraged in the next section to finally provide concrete lower bounds on the running time of $\mathrm{Inv}(f)$-SAT for pSDI-operations f.

For a partial operation f, say of arity k, and a sequence of tuples $t_1, \ldots, t_k$, we say that $f(t_1, \ldots, t_k)$ is a *vacuous application* if $f(t_1, \ldots, t_k)$ is either undefined or $f(t_1, \ldots, t_k) \in \{t_1, \ldots, t_k\}$. If $f(t_1, \ldots, t_k)$ is defined and $f(t_1, \ldots, t_k) \notin \{t_1, \ldots, t_k\}$ we call $f(t_1, \ldots, t_k)$ a *non-vacuous application.*

Definition 6.4. Let $R \subseteq \{0,1\}^n$ be a relation and F a set of Boolean partial operations. A *padding* of R with respect to F is an $(n+m)$-ary relation $\mathcal{F}_R$ such that (1) $\mathrm{Proj}_{1,\ldots,n}(\mathcal{F}_R) = R$, (2) $|\mathcal{F}_R| = |R|$, and (3) $\mathcal{F}_R \in \mathrm{Inv}(F)$. A *universal padding formula* for $n \geq 1$ with respect to F is an $(n+m)$-ary relation $\mathcal{U}P_F$ which (1) is a padding of the relation $\{0,1\}^n$ and (2) $f(t_1, \ldots, t_{\mathrm{ar}(f)})$ is a vacuous application for every partial operation $f \in F$ and every sequence of tuples $t_1, \ldots, t_{\mathrm{ar}(f)} \in \mathcal{U}P_F$.

Note that if R is a relation and f a k-ary partial operation such that $f(t_1, \ldots, t_k)$ is a vacuous application for every sequence $t_1, \ldots, t_k \in R$, then $R \in \mathrm{Inv}(F)$. In particular this implies that $\mathcal{U}P_F \in \mathrm{Inv}(F)$ for every universal padding formula $\mathcal{U}P_F$ of F. Also, critically, if $\mathcal{U}P_F$ is an $(n+m)$-ary universal padding formula for a set of partial operations F, and R is an n-ary relation, then the relation $R'(x_1, \ldots, x_n, y_1, \ldots, y_m) \equiv R(x_1, \ldots, x_n) \wedge \mathcal{U}P_F(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a padding formula for R. Hence, a universal padding formula can be viewed as a blueprint which can be applied to obtain a concrete padding formula for any relation. It is known that if F contains no total operation, then a universal padding formula can be constructed using a universal hash family [41].

Lemma 6.5. *Let F be a finite set of partial operations such that the only total functions in $[F]_s$ are projections. For every $n \geq 1$ there exists an $(n+m)$-ary universal padding formula $\mathcal{U}P_F$ such that $m \leq c \cdot n$, for a constant c depending on F.*

Proof. See Lagerkvist & Wahlström [41, Lemma 35]. □

A quick note is in place on the role of universal padding formulas in obtaining lower bounds for $\mathrm{Inv}(F)$-SAT, when F is a finite set of partial operations. Note that in a standard "gadget" reduction from CNF-SAT to some problem SAT(Γ), one would introduce some number of local variables for every clause of the input, to create an equivalent output formula that only uses constraints from Γ. The existence of padding formulas does allow us to do this for $\mathrm{Inv}(F)$-SAT, but for lower bounds under SETH this is not useful since we have no control over the number of additional variables created this way. However, the universality property of universal padding formulas allow us to *reuse* the padding variables between different constraints, to produce an output which only has $n + m = O(n)$ variables in total. The details are given in the next section (and an overview was given in Section 2.3) but first we investigate concrete values of the constant c for specific operations.

Lemma 6.6. *Let $R(x_1, \ldots, x_n, y_1, \ldots, y_m)$ be a padding formula for $\{0,1\}^n$, where each y_i is a a parity bit over $\{x_1, \ldots, x_n\}$ chosen uniformly at random. Then the following hold.*

1. *For the partial 2-edge operation, $R(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a universal padding formula with probability at least $1 - \varepsilon$ if $m \geq 3.82n + O(\log(1/\varepsilon))$.*
2. *For any minimal non-trivial operation at level $k = 3$, excepting the total operation 3-NU, $R(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a universal padding formula with probability at least $1 - \varepsilon$ if $m \geq 2n + O(\log(1/\varepsilon))$.*
3. *For the partial k-NU operation, $k \geq 4$, and for any operation weaker than it, $R(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a universal padding formula with exponentially small failure probability if $m = \Omega(\frac{\log k}{k} n)$.*

Proof. For the partial 2-edge operation, the result was shown in Section 2.3. The rest of the statements follow a similar line of argumentation, adapted to the various operations. We start with background claims.

Claim 1. *Let f be a partial operation. There are $(|\text{domain}(f)|)^n$ sequences $(t_1, \ldots, t_{\text{ar}(f)})$ of tuples in $\{0,1\}^n$ such that $f(t_1, \ldots, t_{\text{ar}(f)})$ is defined.*

Proof. For every argument $i \in [n]$, we choose which element from domain(f) the tuple $(t_1[i], \ldots, t_{\text{ar}(f)}[i])$ will correspond to. Every such choice results in a distinct sequence of tuples. □

When f is a pSDI-operation, we can reduce the relevant cases by a factor of 2^n. Say that two tuples $(t_1, \ldots, t_r)$ and $(t'_1, \ldots, t'_r)$ over $\{0,1\}^n$ are *equivalent* if, for every $i \in [n]$, either $(t_1[i], \ldots, t_r[i]) = (t'_1[i], \ldots, t'_r[i])$ or the two sequences are complementary. Note that this defines equivalence classes of precisely 2^n sequences. We show that only the equivalence class is relevant for XOR-padding.

For a tuple $(t_1, \ldots, t_r) \in (\{0,1\}^n)^r$ and a set $S \subseteq [n]$, by *padding the tuple with S* we refer to adding a position $n+1$ to each tuple in the sequence, where $t_i[n+1] = \bigoplus_{j \in S} t_i[j]$ for every $i \in [r]$. Write $y_S(t) = \bigoplus_{i \in S} t[i]$.

Claim 2. *Let f be a pSDI-operation with $\text{ar}(f) = r$ and let $y = \bigoplus_{i \in S} x[i]$ be a parity bit with index set S. Let $(t_1, \ldots, t_r)$ and $(t'_1, \ldots, t'_r)$ be equivalent sequences of tuples from $\{0,1\}^n$. After padding the tuples with S, either both of $f(t_1, \ldots, t_r)$ and $f(t'_1, \ldots, t'_r)$ are defined or both are undefined.*

Proof. We assume that $f(t_1, \ldots, t_r)$ (and hence $f(t'_1, \ldots, t'_r)$) are defined, as otherwise the statement is trivial. We claim that either $y(t_i) = y(t'_i)$ for every $i \in [r]$, or $y(t_i) = 1 - y(t'_i)$ for every $i \in [r]$. Indeed, for every position $j \in [n]$, either $t_i[j] = t'_i[j]$ for every $i \in [r]$, or $t_i[j] = 1 - t'_i[j]$ for every $i \in [r]$, by the definition of tuple equivalence. Hence $y(t_i) = y(t'_i)$ if S contains an even number of positions of the latter type, and $y(t_i) = 1 - y(t'_i)$ otherwise, for every $i \in [r]$. Since f is self-dual, in both cases either both of $f(y(t_1), \ldots, y(t_r))$ and $f(y(t'_1), \ldots, y(t'_r))$ are defined, or neither is. □

Fix a pSDI-operation f from the lemma statement, and let $|\text{domain}(f)| = 2k+2$. Enumerate the tuples t of domain(f) for which $f(t) = 0$ as $z_0, \ldots, z_k$, where z_0 is the all-zero tuple. Let the *type* of position $j \in [n]$ of a tuple $(t_1, \ldots, t_r)$ over $\{0,1\}^n$ be the index $i \in \{0, \ldots, k\}$ such that $(t_1[j], \ldots, t_r[j])$ and z_i are either equal or complementary (and note that the type is invariant up to the tuple equivalence we defined). Given a tuple $(t_1, \ldots, t_r)$ over $\{0,1\}^n$ and a set $S \subseteq [n]$, define $I_S(t_1, \ldots, t_r) \subseteq [k]$ as the set of values $i \in [k]$ such that an odd number of positions $j \in S$ are of type i in $(t_1, \ldots, t_r)$. Then I_S is an invariant of the tuple equivalence classes; we claim that I_S determines whether $f(y_S(t_1), \ldots, y_S(t_r))$ is defined. Indeed, let $y_S(t_1) = b$; then $y_S(t_i) \neq b$ for $i \in [r]$ if and only if S indexes an odd number of positions $j \in [n]$ such that $t_1[j] \neq t_i[j]$, which is equivalent to I containing an odd number of types p such that $z_p[1] \neq z_p[i]$. This furthermore determines whether $f(y_S(t_1), \ldots, y_S(t_r))$ is defined, since f is self-dual.

The proof can now be finished, with case-specific constant, using the probabilistic method. There are $(k+1)^n$ types for well-defined applications of f, of which almost all are non-vacuous. There is a case-specific probability p_f that a given non-vacuous application of f remains defined after padding with a random column y_S. Therefore, with the correctly chosen value of m, after m randomly chosen padding columns, on expectation there are at most

$$(k+1)^n p_f^m < \varepsilon$$

tuples $(t_1, \ldots, t_r)$ such that f remains non-vacuous. By Markov's inequality, the probability that the actual number is non-zero is then at most ε. We proceed with computing p_f and m.

1. When f is the partial 2-edge operation, i.e., partial Maltsev, then we have $k = 2$ and the constant $p_f = 3/4$ and the constant factor in $m \approx 3.82n + O(\log 1/\varepsilon)$ were computed in Section 2.3. The final $O(\log(1/\varepsilon))$ part is obvious.

2. Let f be an operation at level $k = 3$, except 3-NU, and let $r = \mathrm{ar}(f)$. By the structure of argument padding among pSDI-operations of Lemma 4.14, we have $r \geq 4$, and there exist three arguments of f, say 1–3, such that projecting f onto those three arguments yields the 3-NU operation, and $|\mathrm{domain}(f)| = 8$. Let $z_1, \ldots, z_3$ be the three distinct non-constant tuples in $\mathrm{domain}(f)$ such that $f(z_i) = 0$. An application $f(t_1, \ldots, t-r)$ is non-vacuous only if all types 1–3 are present, thus the set I_S is a uniformly random subset of $\{1, 2, 3\}$. Furthermore, if $|I_S| \leq 1$, then $f(y_S(t_1), \ldots, y_S(t_r))$ remains defined. We claim that in every other case, f becomes undefined. Let $i > 3$, $i \leq r$ be a position such that $(z_1[i], \ldots, z_3[i])$ has Hamming weight 2 or 3; note that this must exist since f is non-trivial and not equivalent to 3-NU. In both cases, it is easy to check that any case where $|I_S| > 1$ means that $f(y_S(t_1), \ldots, y_S(t_4))$ is undefined, already due to positions 1–3 and i.

Hence $p_f = 1/2$, there are at most 4^n types of non-vacuous applications of f, and we get $m = 2n + O(\log 1/\varepsilon)$.

3. For partial k-NU, $k \geq 4$, there are at most $(k+1)^n$ types of non-vacuous applications of f, and for $S \subseteq [n]$, $f(y_S(t_1), \ldots, y_S(t_r))$ is defined if and only if $|I_S| \leq 1$ or $|I_S| \geq k - 1$. Similarly, all types 1–k must be present in $(t_1, \ldots, t_k)$, or otherwise the application is vacuous. Thus $p_f = (2k+2)/2^k$ and we solve

$$(k+1)^n \left(\frac{2k+2}{2^k}\right)^{cn} = 1$$

for

$$c = \frac{\log(k+1)}{\log(2^k/(2k+2))} = \frac{\log(k+1)}{k - 1 - \log(k+1)} = \Theta((\log k)/k).$$

Adding any $O(n)$ further bits takes care of the margin of error. □

We remark that with a padding strategy other than simple parity bits, a significantly lower scaling ratio may be possible for the partial k-universal operations. However, the advantage of padding with parity bits is that the padding can be efficiently inverted, allowing for efficient extension oracles for the padded relation.

6.4 SETH-based lower bounds for inv(F)-SAT

We now use the bounds obtained above to obtain lower bounds for $\mathrm{Inv}(F)$-SAT in the extension oracle model.

Lemma 6.7. *Let $\mathcal{U}P_F$ be an $(n+m)$-ary universal padding formula via the construction in Lemma 6.6. Let $R = \{0,1\}^k \setminus \{t\}$ for a k-ary tuple $t \in \{0,1\}^k$. Then there is a polynomial-time extension oracle for $R(x_1, \ldots, x_k) \wedge \mathcal{U}P_F(x_1, \ldots, x_n, y_1, \ldots, y_m)$.*

Proof. Let $\alpha : X \to \{0,1\}$, $X \subseteq \{x_1, \ldots, x_k, y_1, \ldots, y_m\}$, be a partial truth assignment. We need to show that we can decide if α is consistent with $R(x_1, \ldots, x_k) \wedge \mathcal{U}P_F(x_1, \ldots, x_n, y_1, \ldots, y_m)$ in polynomial time. First, we check whether α is consistent with the constraint $R(x_1, \ldots, x_k)$, which is easy to do due to the representation of R. Second, recall that there for each y_i exists an index set S_i such that $y_i = \bigoplus_{s \in S_i} x_s$. Hence, the partial assignment α together with $R(x_1, \ldots, x_k) \wedge \mathcal{U}P_F(x_1, \ldots, x_n, y_1, \ldots, y_m)$ induces a system of linear equations over GF(2) where the unknown variables are those unassigned by α. We may thus solve this system and check whether it has any solution f where $f[i] \neq t[i]$ for some $i \in [k]$. □

Theorem 6.8. *Let F be a set of partial operations, and set $m \geq cn + \log n$ such that a random parity-padded formula $\mathcal{U}P_F(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a universal padding formula with high probability. Then $\mathrm{Inv}(F)$-SAT cannot be solved in time $O^*(2^{(1/(c+1)-\varepsilon)n})$ for any $\varepsilon > 0$, assuming the randomised version of the SETH is true. In particular, we have the following lower bounds for specific problems:*

(1) *2-EDGE-SAT cannot be solved in $O(2^{(c-\varepsilon)n})$ time for any $\varepsilon > 0$, where $c \approx 1/4.82$; i.e., $c(\mathrm{Inv}(\phi)) > 1.1547$.*

(2) *3-EDGE-SAT cannot be solved in $O(2^{(c-\varepsilon)n})$ time for any $\varepsilon > 0$, where $c = 1/3$; i.e., $c(\mathrm{Inv}(e_3)) > 1.2599$.*
(3) *4-NU-SAT, being the smallest among the classes that contains 3-SAT, cannot be solved in $O^*(c^n)$ time for any $c \leq 1.1696$.*
(4) *For $k \geq 4$, k-NU-SAT cannot be solved in $O(2^{(c-\varepsilon)n})$ time for any $\varepsilon > 0$, where $c = 1 - \Theta(\frac{\log k}{k})$, and the same bound holds for the harder problems k-EDGE-SAT and k-UNIVERSAL-SAT.*

Proof. Let $\mathcal{F}$ be a CNF-SAT instance on variable set X, $|X| = n$, and compute a random padding formula $\mathcal{U}P_F(x_1, \ldots, x_n, y_1, \ldots, y_m)$, with m as stated. We assume that the construction is successful, i.e., that the resulting relation is a universal padding formula with respect to F. For every clause in the input, defined on a tuple of variables $(x_{i_1}, \ldots, x_{i_r})$, let $R(x_{i_1}, \ldots, x_{i_r})$ be the corresponding relation, and let $R'(x_{i_1}, \ldots, x_{i_r}) \wedge \mathcal{U}P_F(x_1, \ldots, x_n, y_1, \ldots, y_m)$ be the relation as in Lemma 6.7 (up to the ordering of variables). Note that we do not need to explicitly enumerate the tuples in this relation, since we may simply provide the extension oracle proven to exist in Lemma 6.7. Then the output is a conjunction of $\mathrm{Inv}(F)$-SAT relations, with a polynomial-time extension oracle for each one, and the resulting instance is equivalent to $\mathcal{F}$. Since the output instance has $n + m = (c + 1) \cdot n$ variables, an algorithm solving $\mathrm{Inv}(F)$-SAT faster than the time stated would imply an improved algorithm for CNF-SAT. The bounds for specific problems follow from the bounds for universal padding formulas computed in Lemma 6.6. In particular, the bound 1.1696 for 4-NU is calculated through $c = \log_2(5)/(3 - \log_2(5))$; and the precise lower bound for $(k + 1)$-NU, containing k-SAT, is time $2^{1-\frac{\log_2(k+2)}{k}}$. □

Finally, we note that the convergence of the lower bounds for k-NU-SAT towards 2^n, assuming SETH, is at a slower rate than the upper bounds for the best known algorithms for k-SAT, which scale as $c_k \leq 2 - \Theta(1/k)$ [32]. There are also significant differences in problem model (finite language versus infinite language, and concrete constraints versus extension oracles). It would be interesting to improve these results, to either improve the convergence rate or provide bounds in some explicit representation model, assuming SETH.

Section summary. We have proven concrete lower bounds under SETH for sign-symmetric SAT problems. In the process, we have identified 2-EDGE-SAT as the first SAT problem admitting both a non-trivial upper bound and a non-trivial lower bound via the SETH. We also gave a lower bound subject to the Subset Sum problem, which as remarked is strong evidence that the $O^*(2^{\frac{n}{2}})$ algorithm from Theorem 5.2 is the best we could reasonably hope for.

7 DISCUSSIONS AND CONCLUSIONS

We have investigated the structure of constraint languages under fine-grained reductions, with a focus on sign-symmetric Boolean languages, and applied the results to an analysis of the time complexity of NP-hard SAT problems, in a general setting.

The structural analysis uses an algebraic connection to analyse constraint languages via their partial polymorphisms. Thereby the structural conclusions are relevant for any problem that takes as input a constraint formula over some fixed constraint language, under just a few assumptions: (1) that the constraints in the formula are "crisp" rather than soft, and are required to all be satisfied (as opposed to problems such as MAX-SAT, where a feasible solution may falsify some constraints); (2) that there are no structural restrictions of the formula itself (e.g., no bounds on the number of occurrences per variable); and (3) that the constraint language is sign-symmetric, i.e., allows the free application of negated variables and the use of constants in constraints. Thus it naturally applies to SAT(Γ) problems, but would also

be relevant for the analysis of problems such as #SAT and optimisation problems, or even parameterized problems such as Local Search SAT(Γ) – is there a solution within distance k of a given non-satisfying assignment t?

Structural results. The expressive power of sign-symmetric languages is characterised by the restricted partial polymorphisms in this article referred to as pSDI-operations. We characterise the structure of all minimal non-trivial pSDI-operations, and find that they are organised into a hierarchy, whose levels correspond to the problem complexity, with close connections to being able to express the k-SAT languages. Moreover, we described the weakest and strongest operations on each level. We find that particular families of pSDI-operations correspond to partially defined versions of well-known algebraic conditions from the study of CSPs; in particular, the strongest operation at each level k corresponds to the k-NU condition. Finally, we also give a result in the "vertical" direction of the hierarchy, giving a simple characterisation of languages not preserved by the partial k-NU operation for any k. By the above discussion, this result should be of interest also for other inquiries.

Complexity of SAT(Γ) problems. We apply our results to an analysis of the fine-grained time complexity of SAT(Γ) for sign-symmetric languages, under SETH. We consider previously studied languages with improved algorithms – i.e., such that SAT(Γ) can be solved in time $O^*(c^n)$ for some $c < 2$ – and find that they correspond well to particular classes of the hierarchy. Conversely, every known language Γ such that SAT(Γ) is SETH-hard – i.e., admits no improved algorithm assuming SETH – lives entirely outside of the hierarchy. We also show the feasibility of giving improved algorithms whose correctness relies only and directly on the above-mentioned pSDI-operations, by showing that known algorithmic strategies such as fast matrix multiplication and (conjecturally) fast local search can be extended to work for such classes.

Finally, we give complementary lower bounds – for every invariant f as above, there is a constant c_f such that Inv(f)-SAT cannot be solved in $O^*(c^n)$ time for any $c < c_f$, assuming SETH. These results are arguably the first of their kind; every previously known concrete lower bound under SETH has either been for showing that a problem admits no non-trivial algorithm, or has been applied to problems analysed under more permissive parameters such as treewidth. In particular, 2-edge-SAT is the first SAT problem which simultaneously has non-trivial upper and lower bounds on the running time under SETH.

7.1 The abstract problem and polynomial-time connections

Finally, let us make a short detour to consider what we may call the abstract problem. We have noted that for every Boolean pSDI-operation f, there is a set of equational conditions that characterise f, similarly to definitions of varieties in universal algebra, and for every larger domain D, these conditions will uniquely determine a partial operation over the domain D. Furthermore, these conditions are preserved under taking powers of the domain, which we have exploited for particular cases of Inv(f)-SAT and Inv(f)-CSP to reduce input instances to instances of polynomial-time solvable problems on exponentially many variables.

These polynomial-time problem will in general be search problems, like CSPs, and will be preserved by the same type of operation f, but have a fixed number of variables d and with an unbounded domain size n. Let us refer to this as the *abstract* Inv(f)*-problem*. The question can be raised, for which pSDI-operations f does such a problem allow improved polynomial-time algorithms?

We refrain from phrasing the question formally, because the polynomial-time complexity may be strongly affected by details such as constraint representation, but we note that the class of problems defined this way, unlike the original problems SAT(Γ), contain several problems conjectured *not* to have such an improvement.

First, we note that every constraint of arity less than d is preserved by the k-NU-type partial operation with $k \geq d$. This in particular includes the k-hyperclique problem for $(k-1)$-uniform hypergraphs, which has been conjectured not to be solvable in time $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$ and $k > 3$ [45]. Thus the abstract d-NU problem does not admit an improved algorithm for $d > 3$ under this conjecture.

Second, it can be verified that the problem of finding a zero-weight triangle, under arbitrary large edge weights, if viewed as a single constraint of arity $d = 3$, is preserved by the corresponding 3-universal partial operation. It is known that subject to the 3SUM conjecture, this problem cannot be solved in $O(n^{3-\varepsilon})$ for any $\varepsilon > 0$ [64].

If we restrict ourselves to the minimal non-trivial pSDI-operations f defined for the Boolean domain in this article, this leaves only a small number of concrete problems open under the above conjectures. By the inclusions we have established, any operation f at a level $k > 3$ yields an abstract problem as hard as the k-NU operation. Furthermore, the abstract 3-NU problem does admit an improved algorithm via fast matrix multiplication. It can be easily checked that up to argument permutation, there are only eight distinct pSDI-operations f at level 3 of the hierarchy; and by the above discussion, the easiest and the hardest are (conjecturally) resolved. We consider it an interesting question to investigate the complexity of the problem for these remaining cases.

7.2 Regarding a dichotomy for sign-symmetric SAT problems

Ignoring for the moment the lower bounds discussed in the previous section, the results throughout our article suggest a simple potential dichotomy between NP-complete SAT problems solvable in $O(c^n)$ time for $c < 2$ and SAT problems not solvable in $O(c^n)$ time for any $c < 2$ unless SETH fails. We can formulate this conjecture as follows. To simplify the conjecture we restrict ourselves to the non-uniform model.

Conjecture 7.1. *Let Γ be a possibly infinite sign-symmetric Boolean constraint language such that SAT(Γ) is NP-complete. Then SAT(Γ) admits a non-uniform algorithm with running time in $O(c^n)$ time for $c < 2$ if and only if Γ is preserved by a non-trivial pSDI-operation.*

Note that by Corollary 6.2, the negative direction of this conjecture is already known, up to SETH. It thus remains to consider whether k-universal SAT admits a non-uniform improved algorithm for every k. Furthermore, as discussed in the Introduction, the class of constraints definable as the roots of bounded-degree multivariate polynomials represents an example which by Lemma 4.11 is directly associated with k-universal SAT, and which has an improved algorithm by Lokshtanov et al. [48]. Thus, the above conjecture at least represent a kind of Occam's razor-type extrapolation of least mathematical surprise.

However, at the moment this conjecture seems difficult to settle. An extreme negative result, such as the conclusion that the full problem Inv(f)-SAT admits an improved algorithm only when the abstract Inv(f)-problem does, would by Theorem 5.9 need to refute the sunflower conjecture. A full positive resolution would need to generalise the result of Lokshtanov et al. [48] to apply based only on a weak abstract condition, whereas their present algorithm strongly uses properties specific to polynomials. Intermediate outcomes are of course possible, but would raise further questions of which pSDI-operations f are powerful enough to guarantee the existence of an improved algorithm.

7.3 Future work

The investigations in this article leave several concrete open questions, and significant avenues for future work, regarding all parts of the article. Let us highlight a few.

Structural aspects. For better understanding of the language classes it may be fruitful to investigate restricted classes of relations contained in them, such as symmetric relations, or relations R such that both R and its complement belong to the class. For the former, in addition to the existing description for the 2-edge and 3-edge classes, it is easy to see that the (non-trivial) symmetric relations in the k-NU class are precisely those of arity less than k. For the latter, in addition to the existing result regarding block sensitivity, it is possible to show that the corresponding "double-sided version" of the partial 2-edge operation preserves precisely parity functions. In both cases, it may be interesting to extend such descriptions to further classes.

A more concrete question is regarding the structure of $\mathrm{Inv}(\mathrm{nu}_k)$ for $k > 3$. Assume that $R \in \mathrm{Inv}(\mathrm{nu}_k)$ is an n-ary Boolean relation, which depends on every argument. Is there a non-trivial upper bound on $|R|$?

Extension to CSPs. Many questions remain regarding an extension of the project to CSPs on non-Boolean domains. While the minimal non-trivial pSDI-operations defined in this article do have higher-domain analogues, via polymorphism patterns, and while these analogues do in some cases have useful consequences for the complexity of the corresponding CSP, it is not clear that they are in general the only kind of condition that is relevant for the fine-grained complexity of CSPs. In particular, in the Boolean domain there is a known correspondence between pSDI-operations and sign-symmetric languages. No such correspondence has been shown for CSPs in general.

In a different vein, for higher-domain CSPs there are also classes of NP-hard problems whose time complexity is far better than $O^*(|D|^n)$, e.g., k-Colouring corresponds to a CSP of domain size $|D| = k$ and can be solved in $O^*(2^n)$ time for every k [8]. Arguably, we do not have a good understanding of when this occurs in general, and it is thus not clear whether asking for a $O(c^n)$ time algorithm for $c < |D|$ is necessarily the best starting point. To mitigate some of these technical difficulties one may initially only consider constraint languages whose total polymorphisms are the projections.

Problems. Let us mention a few concrete algorithmic questions. First of all, by Lemma 4.12, symmetric relations defined by Sidon sets are preserved by the 3-universal operation, but they do not seem to be captured by currently known algorithms for problems in this class. Does the language consisting of all such relations admit an improved algorithm?

Another problem is to find a generalisation of the algorithm for constraints defined via bounded-degree polynomials [48], without explicitly using properties specific to polynomials. A different generalisation of this class was considered by the present authors in the form of relations with bounded-degree Maltsev embeddings [42]. Since this properly generalises bounded-degree polynomials, it is natural to ask whether this class admits an improved algorithm.

More broadly, as remarked earlier, the classification of the expressiveness of sign-symmetric constraint languages may be of interest for questions other than just satisfiability. The algorithm for 2-edge-SAT, for instance, can be used to solve the corresponding counting problem, showing that pSDI-operations may be powerful enough also in other settings. Concrete questions to consider here include improved algorithms for the counting problem #SAT(Γ) and the parameterized problem Local search SAT(Γ).

Lower bounds. Can the padding scheme be improved to give better asymptotics with respect to the level k? Recall that the lower bound behaves as a bound of $2 - \Theta((\log k)/k)$, whereas all known algorithmic strategies yield running times of the form $(2 - \Theta(1/k))^n$.

It would also be very interesting to have a SETH-based lower bound in the explicit representation model. As discussed earlier the padding construction is valid also in this representation, but is difficult to efficiently implement since the resulting relations may contain exponentially many tuples with respect to the number of variables.

Acknowledgements

We thank the anonomyous reviewers for several helpful comments and suggestions. The first author has received funding from the Swedish resourch council (VR) under grant 2019-03690.

REFERENCES

[1] N. Alon, A. Shpilka, and C. Umans. On sunflowers and matrix multiplication. *Computational Complexity*, 22(2):219–243, 2013.
[2] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.
[3] R. Alweiss, S. Lovett, K. Wu, and J. Zhang. Improved bounds for the sunflower lemma. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC-2020)*, pages 624–630. ACM, 2020.
[4] N. Bansal, S. Garg, J. Nederlof, and N. Vyas. Faster space-efficient algorithms for subset sum and k-sum. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC-2017)*, pages 198–209. ACM, 2017.
[5] L. Barto, A. Krokhin, and R. Willard. Polymorphisms, and How to Use Them. In A. Krokhin and S. Zivny, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017.
[6] L. Barto, J. Opršal, and M. Pinsker. The wonderland of reflections. *Israel Journal of Mathematics*, 223(1):363–398, Feb 2018.
[7] J. Berman, P. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Varieties with few subalgebras of powers. *Transactions of the American Mathematical Society*, 362(3):1445–1473, 2010.
[8] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.
[9] J. Brakensiek and V. Guruswami. Bridging between 0/1 and linear programming via random walks. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC-2019)*, pages 568–577, New York, NY, USA, 2019. ACM.
[10] K. Bringmann, N. Fischer, and M. Künnemann. A fine-grained analogue of Schaefer's theorem in P: dichotomy of exists^k-forall-quantified first-order graph properties. In *Proceedings of the 34th Computational Complexity Conference (CCC-2019)*, volume 137 of *LIPIcs*, pages 31:1–31:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
[11] A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science (FOCS-2017)*. IEEE Computer Society, 2017.
[12] A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM Journal On Computing*, 36(1):16–27, 2006.
[13] A. A. Bulatov. Constraint satisfaction problems: Complexity and algorithms. *ACM SIGLOG News*, 5(4):4–24, Nov. 2018.
[14] C. Calabro, R. Impagliazzo, and R. Paturi. The complexity of satisfiability of small depth circuits. In *Parameterized and Exact Computation, 4th International Workshop (IWPEC 2009)*, pages 75–85, 2009.
[15] C. Calabro, R. Impagliazzo, and R. Paturi. On the exact complexity of evaluating quantified k-CNF. *Algorithmica*, 65(4):817–827, Apr 2013.
[16] H. Chen, B. M. P. Jansen, and A. Pieterse. Best-case and worst-case sparsifiability of Boolean csps. *Algorithmica*, 82(8):2200–2242, 2020.
[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
[18] M. Couceiro, L. Haddad, V. Lagerkvist, and B. Roy. On the interval of Boolean strong partial clones containing only projections as total operations. In *Proceedings of the 47th International Symposium on Multiple-Valued Logic (ISMVL-2017)*, pages 88–93. IEEE Computer Society, 2017.
[19] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. *ACM Transactions on Algorithms*, 12(3):41:1–41:24, 2016.
[20] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
[21] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. M. Kleinberg, C. H. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, 289(1):69–83, 2002.
[22] P. Erdős and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, s1-35(1):85–90, 1960.
[23] F. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.
[24] F. V. Fomin, S. Gaspers, D. Lokshtanov, and S. Saurabh. Exact algorithms via monotone local search. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2016)*, pages 764–775, 2016.
[25] D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27(1):95–100, 1968.
[26] T. D. Hansen, H. Kaplan, O. Zamir, and U. Zwick. Faster k-sat algorithms using biased-ppsz. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC-2019)*, pages 578–589. ACM, 2019.
[27] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. *SIAM Journal on Computing*, 39(5):1667–1713, 2010.
[28] T. Hertli. 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. *SIAM Journal on Computing*, 43(2):718–729, 2014.
[29] T. Hertli. Breaking the PPSZ barrier for unique 3-sat. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP-2014)*, volume 8572 of *Lecture Notes in Computer Science*, pages 600–611. Springer, 2014.
[30] E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21(2):277–292, Apr. 1974.
[31] P. Idziak, P. Marković, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM Journal on Computing*, 39(7):3023–3037, June 2010.
[32] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001.

[33] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512–530, 2001.

[34] B. M. P. Jansen and A. Pieterse. Optimal sparsification for some binary CSPs using low-degree polynomials. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS-2016)*, volume 58, pages 71:1–71:14, 2016.

[35] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, July 1997.

[36] P. Jonsson, V. Lagerkvist, G. Nordh, and B. Zanuttini. Strong partial clones and the time complexity of SAT problems. *Journal of Computer and System Sciences*, 84:52 – 78, 2017.

[37] P. Jonsson, V. Lagerkvist, and B. Roy. Fine-grained time complexity of constraint satisfaction problems. *ACM Transactions on Computation Theory*, 13(1), 2021.

[38] M. Künnemann and D. Marx. Finding small satisfying assignments faster than brute force: A fine-grained perspective into Boolean constraint satisfaction. In *Computational Complexity Conference*, volume 169 of *LIPIcs*, pages 27:1–27:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[39] V. Lagerkvist. *Strong Partial Clones and the Complexity of Constraint Satisfaction Problems: Limitations and Applications*. PhD thesis, Linköping University, The Institute of Technology, 2016.

[40] V. Lagerkvist and B. Roy. A Preliminary Investigation of Satisfiability Problems Not Harder than 1-in-3-SAT. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS-2016)*, pages 64:1–64:14, 2016.

[41] V. Lagerkvist and M. Wahlström. The power of primitive positive definitions with polynomially many variables. *Journal of Logic and Computation*, 27(5):1465–1488, 2017.

[42] V. Lagerkvist and M. Wahlström. Sparsification of SAT and CSP problems via tractable extensions. *ACM Transactions on Computation Theory*, 12(2), 2020.

[43] V. Lagerkvist, M. Wahlström, and B. Zanuttini. Bounded bases of strong partial clones. In *Proceedings of the 45th International Symposium on Multiple-Valued Logic (ISMVL-2015)*, pages 189–194, 2015.

[44] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC-2014)*, pages 296–303, 2014.

[45] A. Lincoln, V. Vassilevska Williams, and R. Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2018)*, pages 1236–1252, 01 2018.

[46] A. Lincoln and A. Yedidia. Faster random k-cnf satisfiability. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP-2020)*, volume 168 of *LIPIcs*, pages 78:1–78:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[47] D. Lokshtanov, D. Marx, and S. Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2011)*, pages 777–789, 2011.

[48] D. Lokshtanov, R. Paturi, S. Tamaki, R. R. Williams, and H. Yu. Beating brute force for systems of polynomial equations over finite fields. In P. N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2017)*, pages 2190–2202, 2017.

[49] K. Meeks. Randomised enumeration of small witnesses using a decision oracle. In *11th International Symposium on Parameterized and Exact Computation (IPEC-2016)*, pages 22:1–22:12, 2016.

[50] B. Monien and E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10(3):287–295, 1985.

[51] N. Nisan. CREW PRAMs and decision trees. *SIAM Journal On Computing*, 20(6):999–1007, 1991.

[52] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[53] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for *k*-sat. *Journal of the ACM*, 52(3):337–364, 2005.

[54] B. Romov. The algebras of partial functions and their invariants. *Cybernetics*, 17(2):157–167, 1981.

[55] T. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory Of Computing (STOC-1978)*, pages 216–226. ACM Press, 1978.

[56] D. Scheder and J. P. Steinberger. PPSZ for general k-SAT - making Hertli's analysis simpler and 3-SAT faster. In *Proceedings of the 32nd Computational Complexity Conference (CCC-2017)*, pages 9:1–9:15, 2017.

[57] D. Scheder and N. Talebanfard. Super strong ETH is true for PPSZ with small resolution width. In *Proceedings of the 35th Computational Complexity Conference (CCC-2020)*, volume 169 of *LIPIcs*, pages 3:1–3:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[58] H. Schnoor and I. Schnoor. Partial polymorphisms and constraint satisfaction problems. In N. Creignou, P. G. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints*, volume 5250 of *Lecture Notes in Computer Science*, pages 229–254. Springer Berlin Heidelberg, 2008.

[59] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS-1999)*, pages 410–414, 1999.

[60] N. Vyas and R. R. Williams. On super strong ETH. In *Prooceedings of the 22nd International Conference on Theory and Applications of Satisfiability Testing (SAT-2019)*, volume 11628 of *Lecture Notes in Computer Science*, pages 406–423. Springer, 2019.

[61] M. Wahlström. *Algorithms, measures and upper bounds for satisfiability and related problems*. PhD thesis, Linköping University, TCSLAB - Theoretical Computer Science Laboratory, The Institute of Technology, 2007.

[62] R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357 – 365, 2005. Automata, Languages and Programming: Algorithms and Complexity (ICALP-A 2004).

[63] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC-2012)*, pages 887–898, 2012.
[64] V. V. Williams and R. Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal On Computing*, 42(3):831–854, 2013.
[65] M. Xiao and H. Nagamochi. Exact algorithms for maximum independent set. *Information and Computation*, 255:126–146, 2017.
[66] D. Zhuk. A proof of the CSP dichotomy conjecture. *Journal of the ACM*, 67(5):30:1–30:78, 2020.

A FINDING BOOLEAN ROOTS OF SYSTEMS OF BOUNDED-DEGREE POLYNOMIALS OVER A FINITE FIELD

We now present the adaptation of the algorithm of Lokshtanov et al. [48] to finding Boolean roots for bounded-degree systems over larger finite fields. We recall the statement.

Theorem 3.1. *Let $\mathbb{F}$ be a fixed finite field and $d \in \mathbb{N}$ a degree bound. There is a randomized algorithm that checks whether a given system of multivariate polynomials over $\mathbb{F}$ of degree at most d has a common root in $\{0,1\}^n$ in time $O(c^n_{\mathbb{F},d})$ for some $c_{\mathbb{F},d} < 2$, where n is the number of variables in the system.*

We need a version of their fast evaluation lemma (see [48, Lemma 2.2]) specialized to only producing evaluations in $\{0,1\}^n$. Note that for such evaluations we can assume that the polynomial is multilinear.

Lemma A.1. *Let P be an n-variate multilinear polynomial over $\mathbb{F}$ provided as a sum of distinct monomials. There is an algorithm that runs in time $n^{O(1)}2^n$ and outputs a vector V such that for every $t \in \{0,1\}^n$, $V[t] = P(t_1, \ldots, t_n)$.*

Proof. Write $P(x_1, \ldots, x_n) = P_1(x_1, \ldots, x_{n-1})x_n + P_0(x_1, \ldots, x_{n-1})$ by splitting the list of monomials according to whether each term contains x_n or not. Recursively compute the vectors V_1 respectively V_0 for evaluations of P_1 and P_0 on all assignments to $x_1, \ldots, x_{n-1}$. For the output vector V, for all assignments where $x_n = 0$ simply copy V_0, and for an assignment $t = (t_1, \ldots, t_n)$ with $t_n = 1$ let $V[t] = V_1[(t_1, \ldots, t_{n-1})] + V_0[(t_1, \ldots, t_n)]$. The base case, where $n \leq 1$, is trivial.

Since a sum of distinct multilinear monomials has at most 2^n terms, the time for setting up the split and summing up the vector V from V_0 and V_1 is bounded as $n^{O(1)}2^n$. Hence the time is bounded as $T(n) = 2T(n-1) + n^{O(1)}2^n$, which gives $T(n) = n^{O(1)}2^n$. □

We now present the proof sketch for Theorem 3.1

Proof. The algorithm is the same as Lokshtanov et al. [48], with different parameter settings. We review the result, with suitable adjustments. Let $\mathbb{F} = GF(q)$ be the finite field of order q for some $q = p^k$ and let $d \in \mathbb{N}$. Let I be an input, consisting of a set of variables $X = \{x_1, \ldots, x_n\}$ and a set of polynomials $p_1, \ldots, p_m$ over $\mathbb{F}$, each of degree at most d. For $i \in [m]$, let $p_i(t_1, \ldots, t_n)$ be the result of evaluating p_i under the assignment where $x_j = t_j$ for each $j \in [n]$ (i.e., we are extending p_i to take n arguments, even if most arguments are ignored). The proof of Lokshtanov et al. combines two main ideas. The first is a randomized degree reduction. Let $\alpha \in \mathbb{F}^m$ be a vector chosen uniformly at random. Then for any tuple $t \in \{0,1\}^n$,

$$\Pr_{\alpha}[\sum_{i=1}^{m} \alpha(i)p_i(t_1, \ldots, t_n) = 0] = \begin{cases} 1 & (t_1, \ldots, t_n) \text{ is a satisfying assignment to the input,} \\ 1/q & \text{otherwise.} \end{cases}$$

Furthermore, denote

$$P_\alpha(X) = 1 - \left(\sum_{i=1}^{m} \alpha(i)p_i(X)\right)^{q-1}.$$

Then for any $t \in \{0,1\}^n$, $P_\alpha(t) = 1$ if t is a satisfying assignment, and if not then $P_\alpha(t) = 0$ with probability $1 - 1/q$ (over the choice of α). Furthermore, let $S = \{\alpha_1, \ldots, \alpha_\ell\}$ be a collection of vectors in $\mathbb{F}^m$ and let

$$Q_S(X) = \prod_{i=1}^{\ell} P_{\alpha_i}(X).$$

If each α_i is chosen independently and uniformly at random, then for any non-satisfying assignment t, the probability that $Q_S(t) \neq 0$ is $q^{-\ell}$. On the other hand, $Q_S(X)$ has degree up to $\ell(q-1)d$, whereas the "true" polynomial Q such that $Q(t) = 1$ if t is a satisfying assignment and $Q(t) = 0$ otherwise could require degree up to n. Indeed, if I has a unique satisfying assignment, say $t = (1, \ldots, 1)$, then $Q(X) = \prod_{i=1}^{n} x_i$.

The other main ingredient in the proof is an application of fast parallel evaluation of polynomials [48, Lemma 2.2]. For our purposes, we are using Lemma A.1 here. For some parameter δ, we split the variables as $X = Y \cup Z$, where $|Z| = \lceil \delta n \rceil =: n'$. For the sake of argument, let $\beta \in \mathbb{F}^{2^{n'}}$ and define the "ideal" polynomial

$$R(Y) = \sum_{a \in \{0,1\}^{n'}} \beta(a) Q(Y, a).$$

Then for any assignment to Y which cannot be extended to a satisfying assignment to Q we have $R(Y) = 0$, whereas for any Y which does have such an extension we have $R(Y) \neq 0$ with probability at least $1 - 1/q$. Unfortunately, as above, $R(Y)$ will have degree up to $n - n'$, and simply preparing it as a list of monomials could take 2^n time. Instead, let β be as above and independently at random select $2^{n'}$ sets S_a, each containing ℓ vectors $\alpha_{a,i}$, chosen as above. Then define

$$\tilde{R}(Y) = \sum_{a \in \{0,1\}^{n'}} \beta(a) Q_{S_a}(Y, a).$$

Finally set $\ell = n' + 2$. First, we consider the probability that $\tilde{R}(Y)$ mimics $R(Y)$. Let $t = (t', t'')$ be an assignment to (Y, Z), and first assume that t is a satisfying assignment. Then $Q_{S_{t''}}(t', t'') = 1$, hence $\tilde{R}(t') \neq 0$ with probability $1 - 1/q$. On the other hand, if t is not satisfying then the probability that $Q_{S_{t''}}(t', t'') \neq 0$ is bounded as $q^{-\ell}$, and if no assignment (t', a) for $a \in 2^{n'}$ is satisfying then by the union bound the probability that $\tilde{R}(t') \neq 0$ is bounded by $2^{n'} q^{-n'-2} \leq 1/4$. Hence for every $t' \in 2^{n-n'}$ there is a constant-factor separation in the probability that $\tilde{R}(t') = 0$ based on whether t' can be extended to a satisfying assignment for I or not. Repeating the randomization and evaluation $\Omega(n)$ times will boost the separation to the point that we can tell the difference between satisfiable and unsatisfiable instances I overall.

It remains to consider how to evaluate $\tilde{R}(Y)$ over all points Y in $O^*(c^n)$ time for some $c < 2$. We show only the asymptotics in the exponent, without optimizing constants. The running time is split into two parts. The second, of evaluating $\tilde{R}$ in all points given a description of $\tilde{R}$ as a sum of monomials, is done in $O^*(2^{(1-\delta)n})$ time using Lemma A.1, hence we focus on assembling such a description of $\tilde{R}$. We will collect this as an explicit sum over all $2^{n'}$ terms $\beta(a) Q_{S_a}(Y, a)$. We note that we only need to maintain a list of multilinear monomials since all evaluations are in $\{0,1\}^n$, i.e., at any step of the process we can reduce a list of monomials down to the corresponding multilinear terms. For each t'', the polynomial $Q_S(Y, Z)$ has degree up to $\ell(q-1)d = (1 + o(1))\delta n(q-1)d$. Let $\delta = c/((q-1)d)$ for some $c = O(1)$; in particular such that $\ell(q-1)d < (n - n')/2$. Then, up to lower-order terms there are

$$\binom{n - n'}{\ell(q-1)d} \approx \binom{(1-\delta)n}{(q-1)d\delta n} \leq \binom{n}{cn}$$

multilinear terms over Y from each contribution $\beta(a) Q_{S_a}(Y, a)$. Each such list of monomials can be constructed naively over the description of Q_S, in a way that causes no significant overhead; see Lokshtanov et al. [48] for a discussion on

this. Hence the total time to construct $\tilde{R}$ is upper-bounded asymptotically as $O^*(2^{H(c)n}2^{\delta n})$ where $H(c)$ is the binary entropy function, and the time to evaluate $\tilde{R}$ is $O^*(2^{(1-\delta)n})$. Clearly there is a $c = \Theta(1)$ such that $H(c) \leq 1 - 2\delta$. Then for such a c the total running time of the procedure is $O^*(2^{(1-\delta)n})$ where $\delta > 0$. Hence for every q and d there is an improved algorithm, and in particular if $\mathbb{F}$ is kept fixed, then the procedure scales with d as $O^*(2^{(1-\Theta(1/d))n})$. □